

# High Performance Networking

SC2002 Tutorial M12

18 November 2002

Phillip Dykstra

Chief Scientist

WareOnEarth Communications Inc.

phil@sd.wareonearth.com

## Motivation

*If our networks are so fast, how  
come my ftp is so slow?*

# Objectives

- Look at current high performance networks
- Learn what is required for high speed data transfer and what to expect
- Fundamental understanding of delay, loss, bandwidth, routes, MTU, windows
- Look at useful tools and what they tell you
- Examine TCP dynamics and TCP's future

Dijkstra, SC2002

3

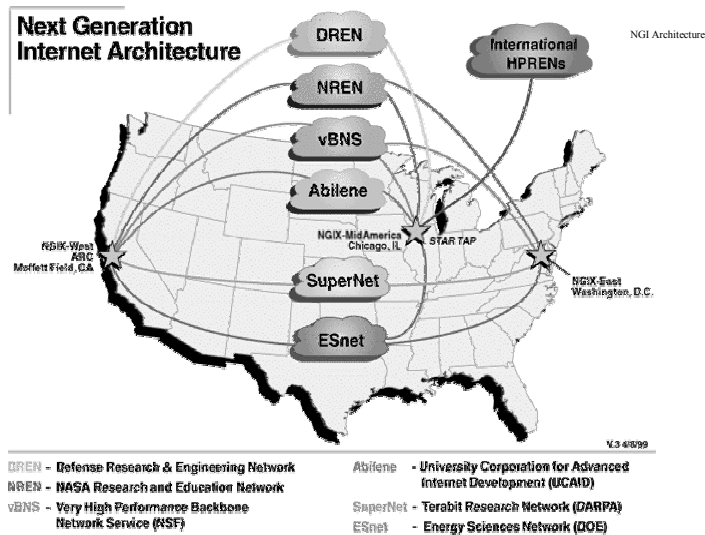
# Unique HPC Environment

- The Internet is being optimized for:
  - millions of users behind low-speed soda straws
  - thousands of high-bandwidth servers serving millions of soda straw streams
- Single high-speed to high-speed flows get little commercial attention

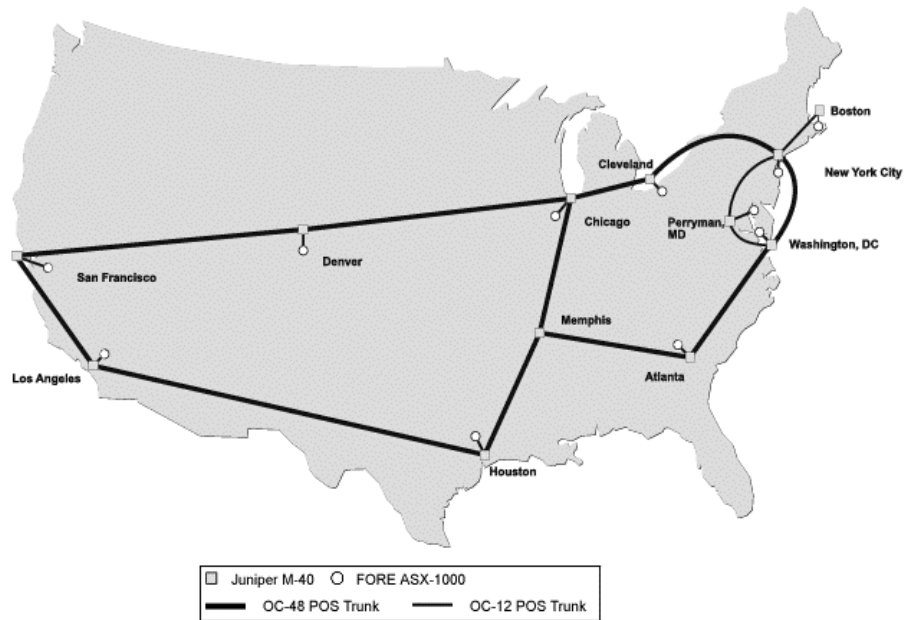
Dijkstra, SC2002

4

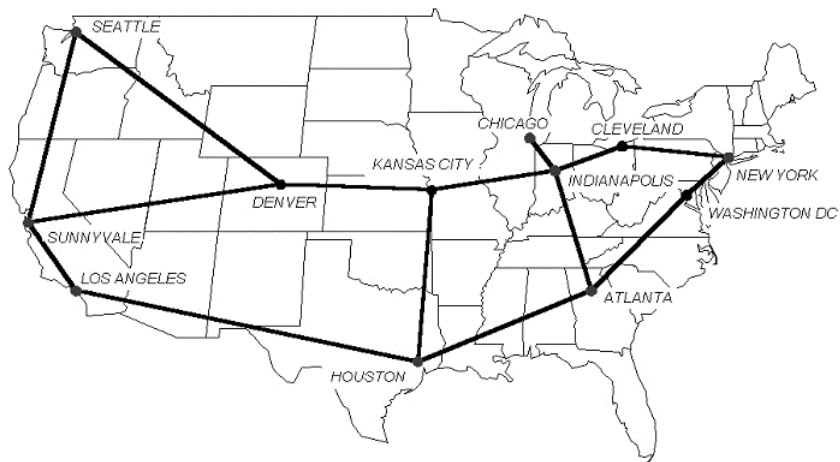
# High Performance Networks in the USA



## vBNS+ Backbone Network Map

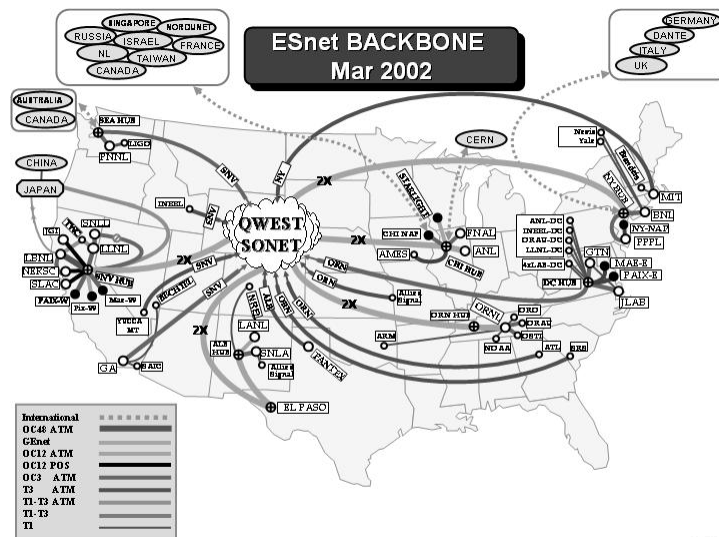


## Abilene Network Backbone - February 2002



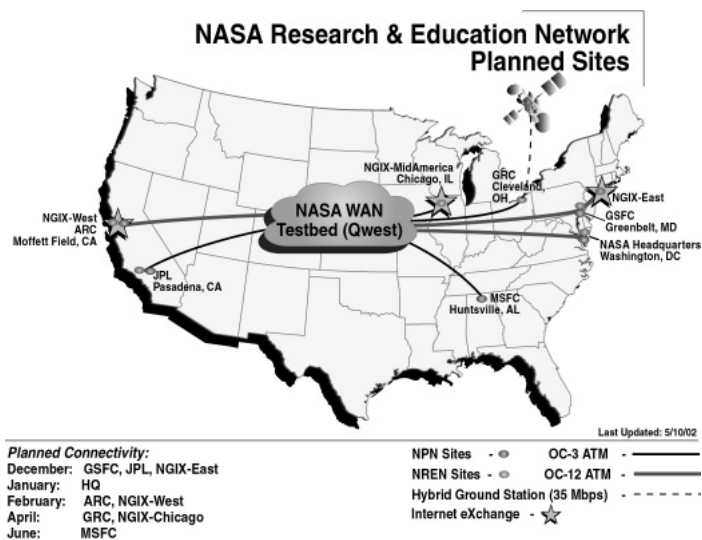
Dykstra, SC2002

8



Dykstra, SC2002

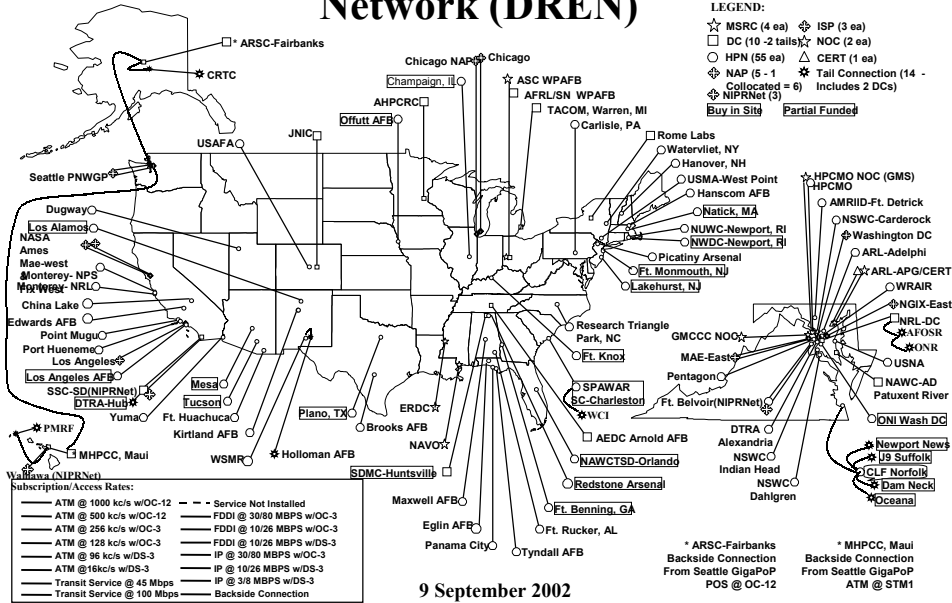
9



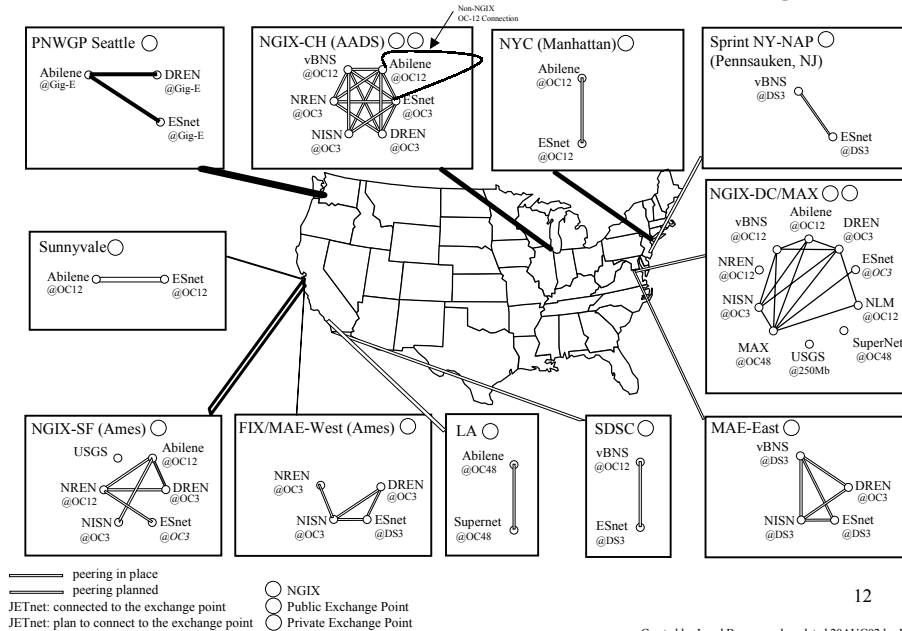
Dykstra, SC2002

10

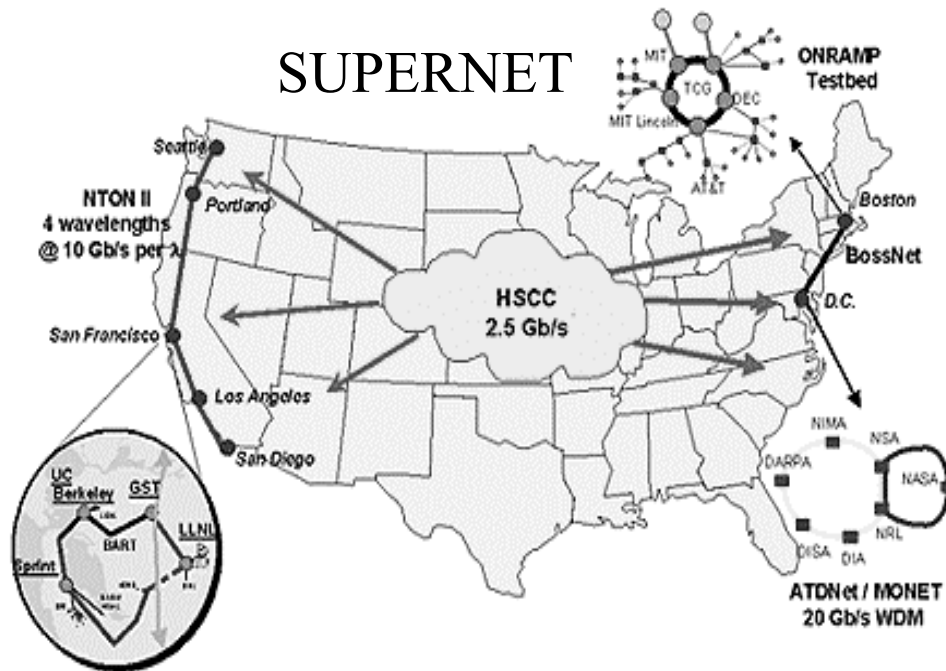
# Defense Research and Engineering Network (DREN)



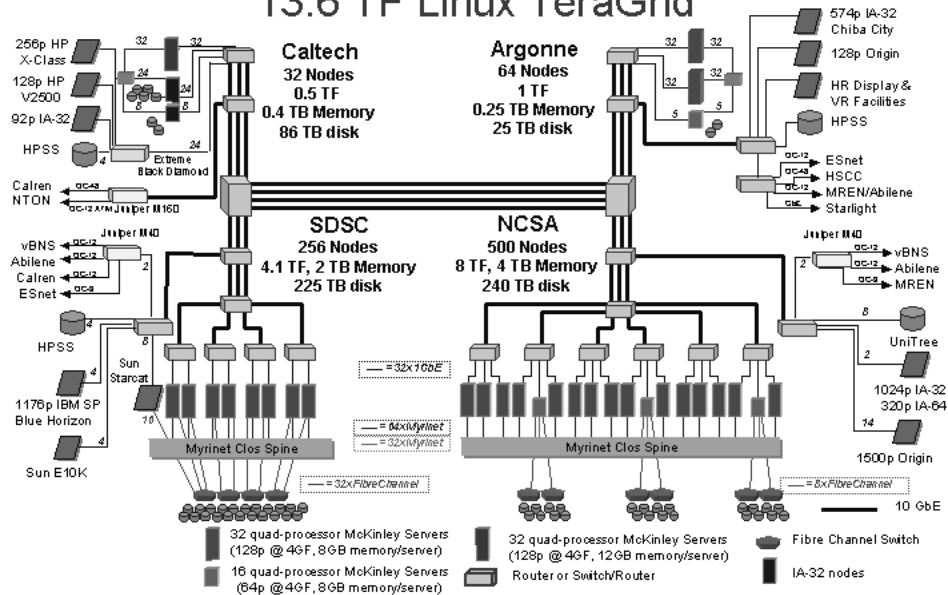
## JETnets Interconnections and Peering



# SUPERNET

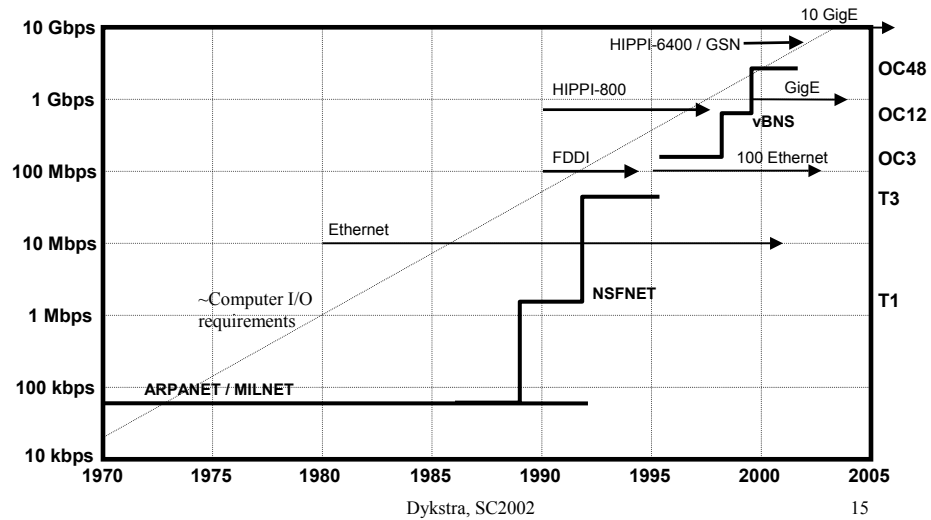


## 13.6 TF Linux TeraGrid



Charlie Catlett (catlett@mcs.anl.gov)

# Network Speeds Over Time

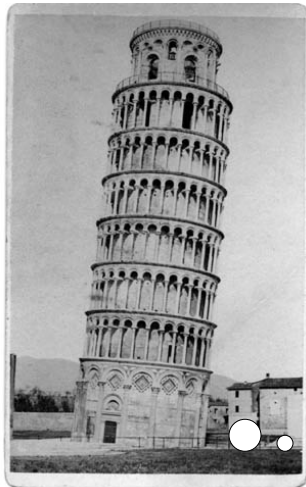


## Delay

a.k.a. Latency



## Capacity High “Speed” Networks



Dykstra, SC2002

OC3  
155 Mbps



DS3  
45 Mbps



17

## Speed of Light in Media

- $\sim 3.0 \times 10^8$  m/s in free space
- $\sim 2.3 \times 10^8$  m/s in copper
- $\sim 2.0 \times 10^8$  m/s in fiber = 200 km / ms  
[100 km of distance = 1 ms of round trip time]

Dykstra, SC2002

18

# Packet Durations and Lengths

## 1500 Byte Packets in Fiber

	<b>Mbps</b>	<b>pps</b>	<b>sec/pkt</b>	<b>length</b>
<b>56k</b>	0.056	4.7	214 ms	42857 km
<b>T1</b>	1.544	129	7.8 ms	1554 km
<b>Eth</b>	10	833	1.2 ms	240 km
<b>T3</b>	45	3750	267 us	53 km
<b>FEth</b>	100	8333	120 us	24 km
<b>OC3</b>	155	13k	77 us	15 km
<b>OC12</b>	622	52k	19 us	3859 m
<b>GigE</b>	1000	83k	12 us	2400 m
<b>OC48</b>	2488	207k	4.8 us	965 m
<b>10GigE</b>	10000	833k	1.2 us	240 m

Dijkstra, SC2002

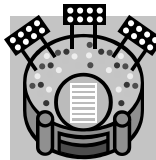
19

## Observations on Packet Lengths

- A 56k packet could wrap around the earth!



- A 10GigE packet fits in the convention center



Dijkstra, SC2002

20

## Observations on Packet Lengths

- Each store and forward hop adds the packet duration to the delay
  - In the old days ( $< 10$  Mbps) such hops dominated delay
  - Today ( $> 10$  Mbps) store and forward delays on WANs are minimal compared to propagation

Dijkstra, SC2002

21

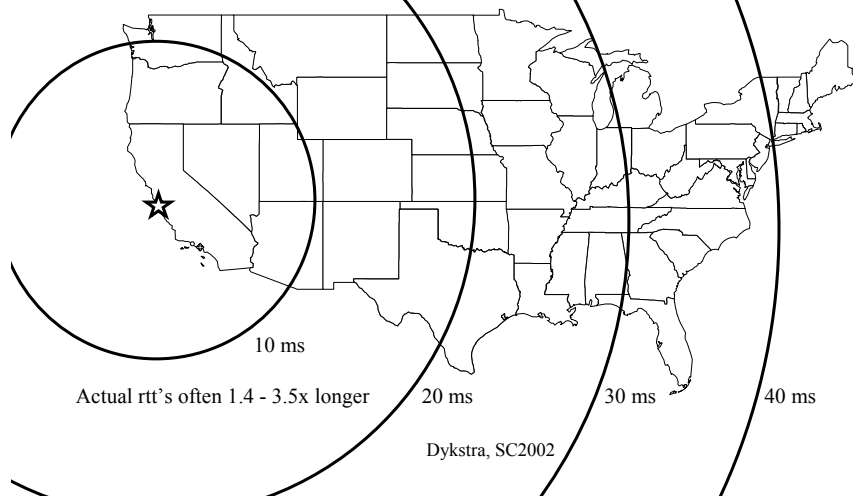
## Observations on Packet Lengths

- ATM cells (and TCP ACK packets) are  $\sim 1/30^{\text{th}}$  as long, 30x as many per second
  - One of the reasons we haven't seen OC48 SAR until now (2002)
- Jumbo Frames (9000 bytes) are 6x longer,  $1/6^{\text{th}}$  as many per second

Dijkstra, SC2002

22

## Light Speed Delay in Fiber



23

## Measuring Delay - Ping

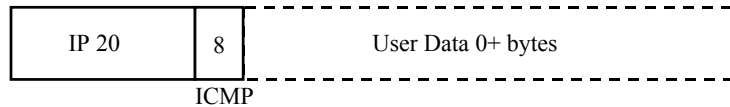
```
% ping -s 56 sgi.com
PING sgi.com (192.48.153.65) from 63.196.71.246 : 56(84) bytes of data.
64 bytes from SGI.COM (192.48.153.65): icmp_seq=1 ttl=240 time=31.6 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=2 ttl=240 time=66.9 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=3 ttl=240 time=33.4 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=4 ttl=240 time=36.7 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=5 ttl=240 time=40.9 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=6 ttl=240 time=104.8 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=7 ttl=240 time=177.5 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=8 ttl=240 time=34.2 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=9 ttl=240 time=31.5 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=10 ttl=240 time=31.9 ms

--- sgi.com ping statistics ---
11 packets transmitted, 10 packets received, 9% packet loss
round-trip min/avg/max = 31.5/58.9/177.5 ms
```

Dijkstra, SC2002

24

## Ping Observations



- Ping packet = 20 bytes IP + 8 bytes ICMP + “user data” (first 8 bytes = timestamp)
- Default = 56 user bytes = 64 byte IP payload = 84 total bytes
- Small pings (-s 8 = 36 bytes) take less time than large pings (-s 1472 = 1500 bytes)

Dijkstra, SC2002

25

## Ping Observations

- TTL = 240 indicates  $255 - 240 = 15$  hops
- Delay variation indicates congestion or system load
- Not good at measuring small loss
  - An HPC network should show zero ping loss
- Depends on ICMP ECHO which is sometimes blocked for “security”

Dijkstra, SC2002

26

## Bandwidth\*Delay Product

- The number of bytes in flight to fill the entire path
- Includes data in queues if they contributed to the delay
- Example
  - 100 Mbps path
  - ping shows a 75 ms rtt
  - $BDP = 100 * 0.075 = 7.5$  million bits (916 KB)

Dijkstra, SC2002

27

## Routes

The path taken by your packets

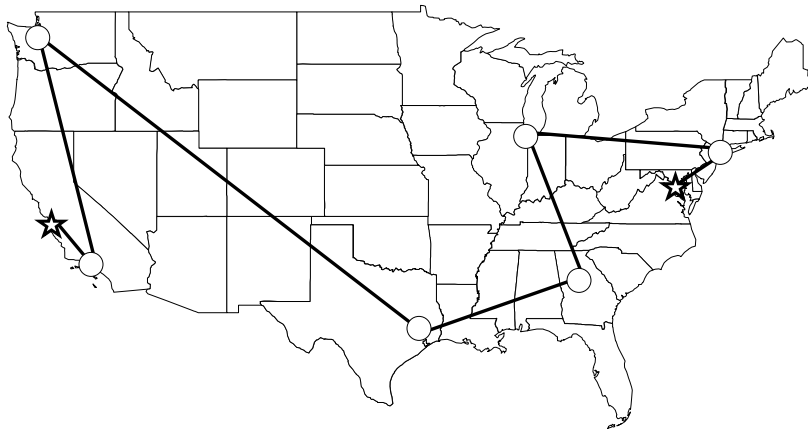
# How Routers Choose Routes

- Within a network
  - Smallest number of hops
  - Highest bandwidth paths
  - Usually ignore latency and utilization
- From one network to another
  - Often “hot potato” routing, i.e. pass to the other network ASAP

Dijkstra, SC2002

29

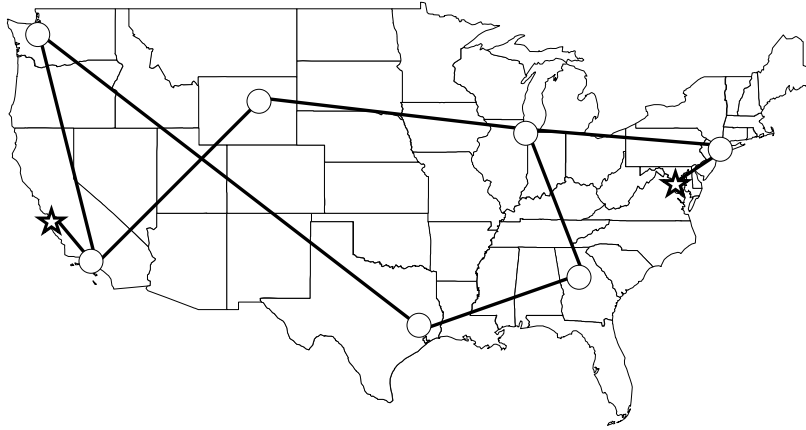
## “Scenic” Routes



Dijkstra, SC2002

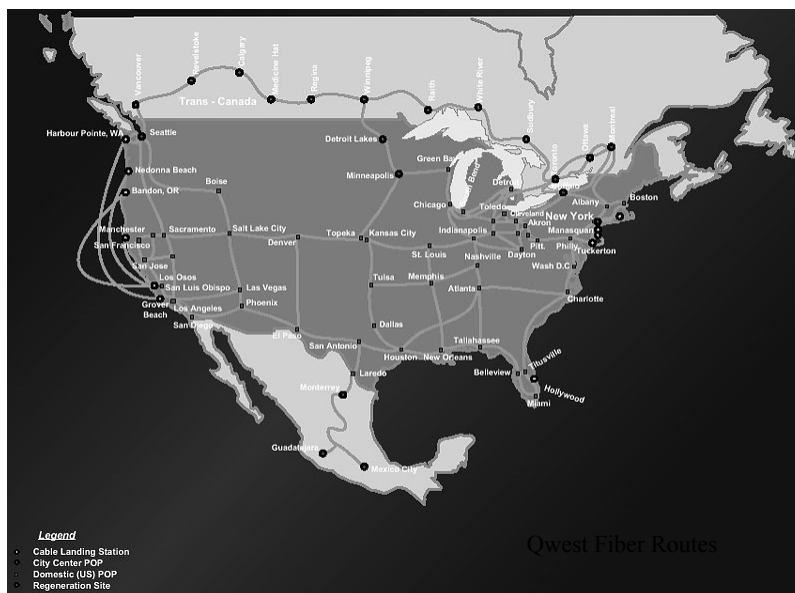
30

# Asymmetric Routes



Dykstra, SC2002

31

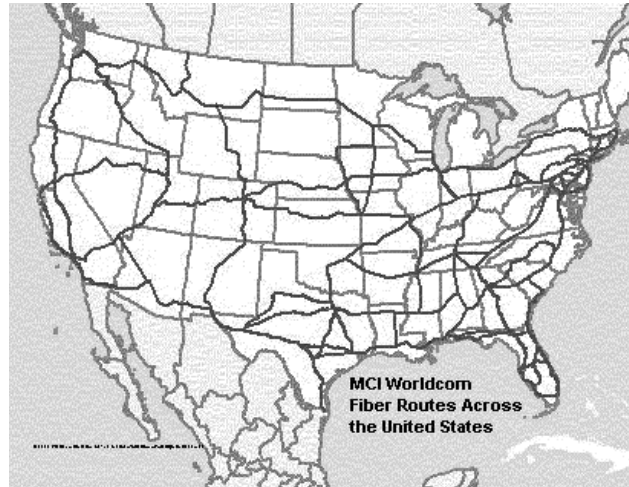


Dykstra, SC2002

32

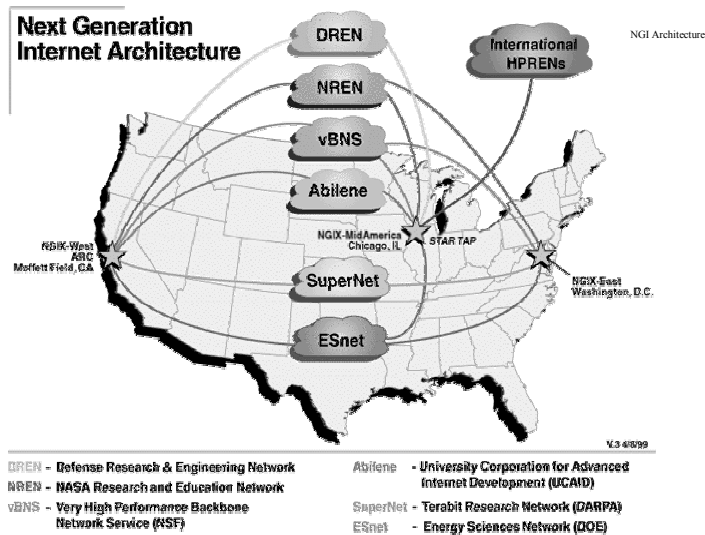


# Worldcom Fiber Routes



Dykstra, SC2002

33

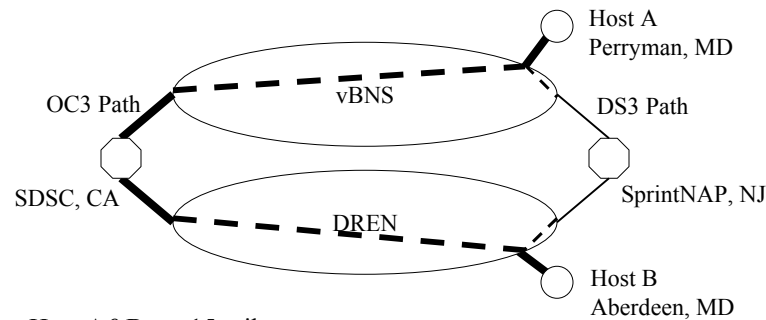


Dykstra, SC2002

34

## Path Performance: Latency vs. Bandwidth

The highest bandwidth path is not always the highest throughput path!



- Host A&B are 15 miles apart
- DS3 path is ~250 miles
- OC3 path is ~6000 miles

*The network chose the OC3  
path with 24x the rtt, 80x BDP*

Dykstra, SC2002

35

## How Traceroute Works

[www.caida.org/outreach/resources/animations/](http://www.caida.org/outreach/resources/animations/)

- Sends UDP packets to ports (-p) 33434 and up, TTL of 1 to 30
- Each router hop decrements the TTL
- If the TTL=0, that node returns an ICMP TTL Expired
- The destination host returns an ICMP Port Unreachable

Dykstra, SC2002

36

# Traceroute Observations

- Shows the return interface addresses of the **forwarding path**
- You can't see hops through switches or over tunnels (e.g. ATM VC's, GRE, MPLS)
- The required ICMP replies are sometimes blocked for "security", or not generated, or sent without resetting the TTL

Dijkstra, SC2002

37

## Matt's Traceroute

[www.bitwizard.nl/mtr/](http://www.bitwizard.nl/mtr/)

```

Matt's traceroute [v0.41]
damp-ssc.spawar.navy.mil Sun Apr 23 23:29:51 2000
Keys: D - Display mode R - Restart statistics Q - Quit

          Packets
Hostname %Loss Rcv Snt Last Best Avg Worst
1.  taco2-fe0.nci.net      0% 24 24 0 0 0 1
2.  nccosc-bgp.att-disc.net 0% 24 24 1 1 1 6
3.  pennsbr-aip.att-disc.net 0% 24 24 84 84 84 86
4.  sprint-nap.vbns.net    0% 24 24 84 84 84 86
5.  cs-hssi1-0.pym.vbns.net 0% 23 24 89 88 152 407
6.  jnl-atl-0-0-0.pym.vbns.net 0% 23 23 88 88 88 90
7.  jnl-atl-0-0-13.nor.vbns.net 0% 23 23 88 88 88 90
8.  jnl-so5-0-0-0.dng.vbns.net 0% 23 23 89 88 91 116
9.  jnl-so5-0-0-0.dnj.vbns.net 0% 23 23 112 111 112 113
10. jnl-so4-0-0-0.hay.vbns.net 0% 23 23 135 134 135 135
11. jnl-so0-0-0-0.rto.vbns.net 0% 23 23 147 147 147 147
12. 192.12.207.22         5% 22 23 98 98 113 291
13. pinot.sdsc.edu        0% 23 23 152 152 152 156
14. ipn.caida.org         0% 23 23 152 152 152 160

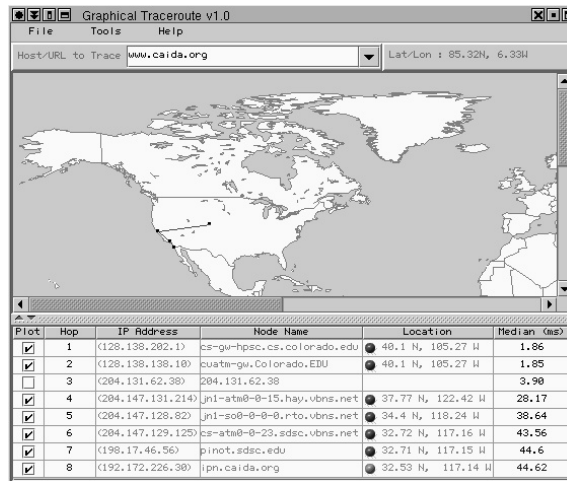
```

Dijkstra, SC2002

38

# GTrace – Graphical Traceroute

[www.caida.org/tools/visualization/gtrace/](http://www.caida.org/tools/visualization/gtrace/)



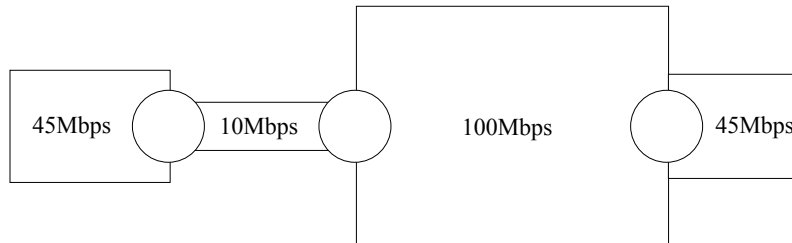
Dijkstra, SC2002

39

## Bandwidth

and throughput

## Hops of Different Bandwidth



- The “Narrow Link” has the lowest bandwidth
- The “Tight Link” has the least **Available** bandwidth
- Queues can form wherever available bandwidth decreases
- A queue buildup is most likely in front of the Tight Link

Dijkstra, SC2002

41



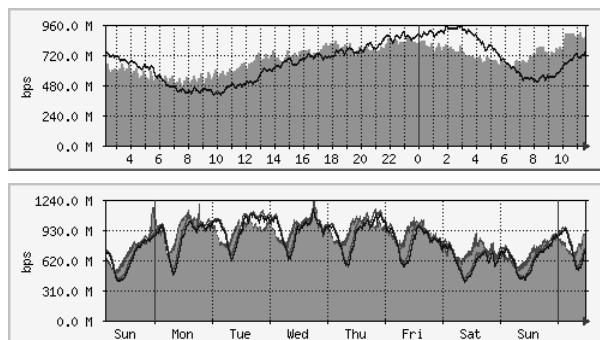
- [www.mrtg.org](http://www.mrtg.org)
- Extremely popular network monitoring tool
- Most common display:
  - Five minute average link utilizations
  - Green into interface
  - Blue out of interface
- RRDTool newer generalized version (same site)

Dijkstra, SC2002

42

# MRTG Example

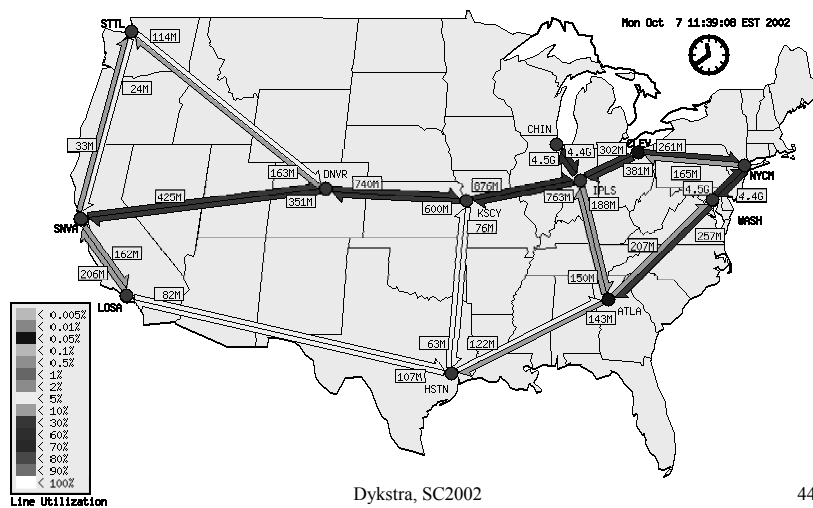
Abilene, Indianapolis to Kansas City, OC48 link, 7 October 2002



Dykstra, SC2002

43

## Abilene “Weather Map”



Dykstra, SC2002

44

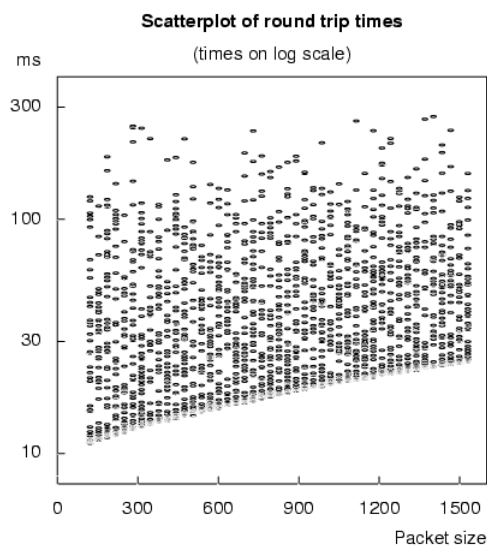
# Throughput Limit

- throughput  $\leq$  **available** bandwidth  
(“tight link” with the minimum unused bandwidth)
  - A high performance network should be lightly loaded ( $<50\%$ )
  - *A loaded high speed network is no better to the end user than a lightly loaded slow one*

Dijkstra, SC2002

45

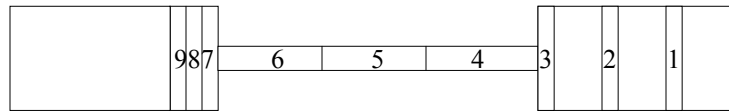
## Bandwidth Estimation – Single Packet



- Larger packets take longer
- Delay from intercept
- Bandwidth from slope

From A. Downey

## Bandwidth Estimation – Multi Packet



- Packet pairs or trains are sent
- The slower link causes packets to spread
- The packet spread indicates the bandwidth

Dijkstra, SC2002

47

## Bandwidth Measurement Tools

- pathchar – Van Jacobson, LBL
  - <ftp://ftp.ee.lbl.gov/pathchar/>
- clink – Allen Downey, Wellesley College
  - <http://rocky.wellesley.edu/downey/clink/>
- pchar – Bruce A. Mah, Sandia/Cisco
  - <http://www.employees.org/~bmah/Software/pchar/>

Dijkstra, SC2002

48



## Bandwidth Measurement Tools

- pipechar - Jin Guojun, LBL
  - <http://www.didc.lbl.gov/pipechar/>
- nettimer - Kevin Lai, Stanford University
  - <http://gunpowder.stanford.edu/~laik/projects/nettimer/>
- pathrate/pathload - Constantinos Dovrolis, Georgia Tech
  - <http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/bwometer.html>

Dijkstra, SC2002

49

## Treno Throughput Test

[www.psc.edu/networking/treno\\_info.html](http://www.psc.edu/networking/treno_info.html)

- Tells you what a good TCP should be able to achieve (Bulk Transfer Capacity)

```
damp-mhpcc% treno damp-pmrf
MTU=8166 MTU=4352 MTU=2002 MTU=1492 .....
Replies were from damp-pmrf [192.168.1.1]
Average rate: 63470.5 kbp/s (55241 pkts in + 87 lost = 0.16%) in 10.03 s
Equilibrium rate: 63851.9 kbp/s (54475 pkts in + 86 lost = 0.16%) in 9.828 s
Path properties: min RTT was 8.77 ms, path MTU was 1440 bytes
```

Dijkstra, SC2002

50

## Treno Observations

- Easy 10 second test, no remote access or receiver process required
- Emulates TCP but doesn't use TCP
  - Problems with host TCP or tuning are avoided
- Does Path MTU Discovery
- Reports rtt and loss rates
- A zero equilibrium result means there was too much packet loss to exit “slow start”

Dijkstra, SC2002

51

## Treno Observations

- Can send ICMP (-i) or UDP (default)
  - ICMP replies (ECHO or UNREACH) could be blocked for “security”
- Routers send ICMP replies very slowly
  - So don't test routers with treno
- ICMP is often rate limited now by hosts
- Port numbers can wrap around (and look like port scans)

Dijkstra, SC2002

52

# TCP Throughput Tests

- `ttcp` – the original, many variations
  - <http://sd.wareonearth.com/~phil/net/ttcp/>
- `nuttcp` – great successor to `ttcp` (recommended)
  - <ftp://ftp.lcp.nrl.navy.mil/u/bill/beta/nuttcp/>
- `Iperf` – great TCP/UDP tool (recommended)
  - <http://dast.nlanr.net/Projects/Iperf/>
- `netperf` – dated but still in wide use
  - <http://www.netperf.org/>
- `ftp` – nothing beats a real application

Dijkstra, SC2002

53

## nuttcp: TCP, 10ms rtt, OC12 path

```
damp-ar1$ nuttcp -r -T10 -w1000 damp-asc2-atm
nuttcp-r: v3.1.9: socket
nuttcp-r: buflen=65536, nstream=1, port=5001 tcp
nuttcp-r: accept from 192.168.131.9
nuttcp-r: send window size = 65536, receive window size = 2048000
nuttcp-r: 604.860 MB in 10.01 real seconds = 61895.32 KB/sec = 507.0464 Mb/s
nuttcp-r: 69593 I/O calls, msec/call = 0.15, calls/sec = 6954.54
nuttcp-r: 0.0user 0.7sys 0:10real 8% 0i+0d 0maxrss 0+0pf 0+0csw

nuttcp-t: v3.1.9: socket
nuttcp-t: buflen=65536, nstream=1, port=5001 tcp -> 140.32.131.13
nuttcp-t: time limit = 10 seconds
nuttcp-t: connect to 192.168.131.13
nuttcp-t: send window size = 2048000, receive window size = 349520
nuttcp-t: 604.860 MB in 10.00 real seconds = 61958.83 KB/sec = 507.5667 Mb/s
nuttcp-t: 9678 I/O calls, msec/call = 1.06, calls/sec = 968.13
nuttcp-t: 0.0user 3.3sys 0:10real 33% 0i+0d 0maxrss 0+0pf 0+0csw
```

Dijkstra, SC2002

54

# Throughput Testing Notes

- Network data rates (bps) are powers of 10, not powers of 2 as used for Bytes
  - E.g. 100 Mbps ethernet is 100,000,000 bits/sec
  - Some tools wrongly use powers of 2 (e.g. ttcp)
- User payload data rates are reported by tools
  - No TCP, IP, Ethernet, etc. headers are included
  - E.g. 100 Mbps ethernet max is 97.5293 Mbps
    - <http://sd.wareonearth.com/~phil/net/overhead/>

Dijkstra, SC2002

55

## Windows

Flow/rate control and error recovery

# Windows

- Windows control the amount of data that is allowed to be “in flight” in the network
- Maximum throughput is one window full per round trip time
- The sender, receiver, and the network each determine a different window size

Dijkstra, SC2002

57

## Window Sizes 1,2,3

Data packets go one way  
ACK packets come back

Dijkstra, SC2002

58

# MPing – A Windowed Ping

[www.psc.edu/~mathis/wping/](http://www.psc.edu/~mathis/wping/)

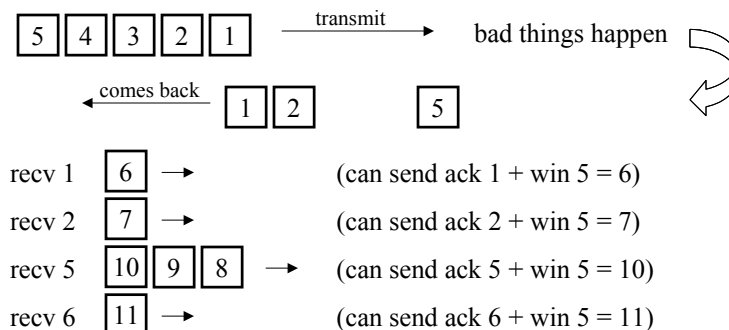
- Excellent tool to view the packet forwarding and loss properties of a path under varying load!
- Sends windows full of ICMP Echo or UDP packets
- Treats ICMP Echo\_Reply or Port\_Unreachable packets as “ACKs”
- Make sure destination responds well to ICMP
- Consumes a lot of resources: use with care

Dijkstra, SC2002

59

## How MPing Works

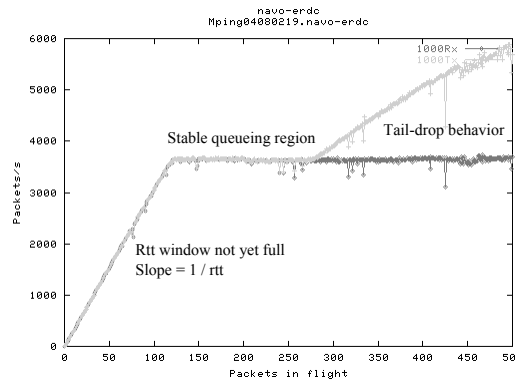
Example: window size = 5



Dijkstra, SC2002

60

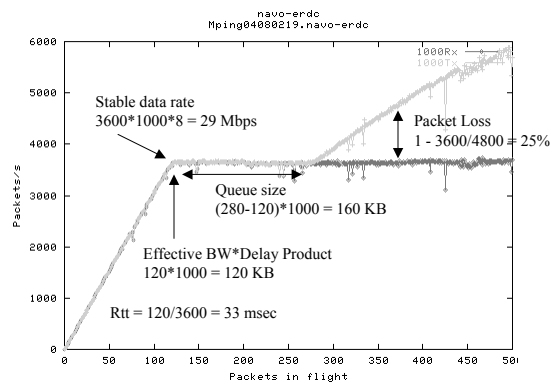
## MPing on a “Normal” Path



Dykstra, SC2002

61

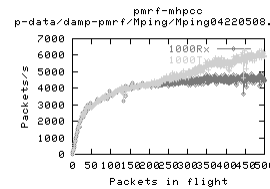
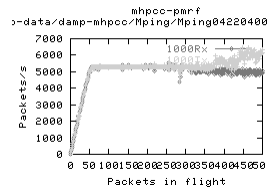
## MPing on a “Normal” Path



Dykstra, SC2002

62

## Some MPing Results #1



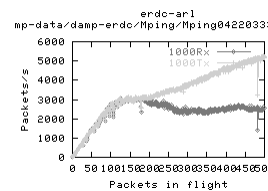
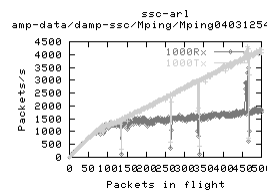
Fairly normal behavior  
Discarded packets are costing  
some performance loss

RTT is increasing as load  
increases  
Slow packet processing?

Dijkstra, SC2002

63

## Some MPing Results #2



Very little stable queueing  
Insufficient memory?  
Spikes from some periodic  
event (cache cleaner?)

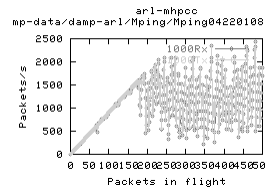
Discarding packets comes at  
some cost to performance  
Error logging?

Dijkstra, SC2002

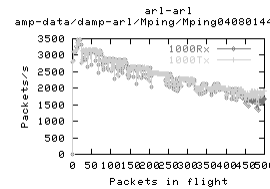
64



## Some MPing Results #3



Oscillations with little loss  
Rate shaping?

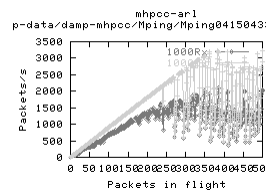


Decreasing performance with  
increasing queue length  
Typical of Unix boxes with  
poor queue insertion

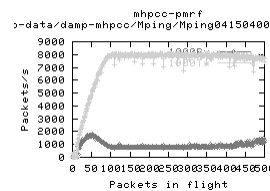
Dijkstra, SC2002

65

## Some MPing Results #4



Fairly constant packet loss,  
even under light load



Major packet loss, ~7/8 or 88%  
Hump at 50 may be duplex problem

*Both turned out to be an auto-negotiation duplex problem  
Setting to static full-duplex fixed these!*

Dijkstra, SC2002

66

# TCP Throughput

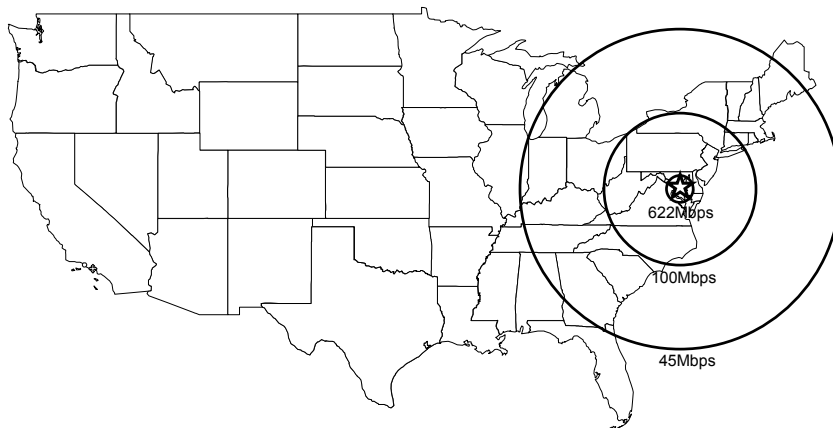
(window/rtt)

- The smallest of three windows determines throughput
  - sbuf, or sender side socket buffers
  - rwin, the receive window size
  - cwin, TCP congestion window
- Receive window (rwin) and/or sbuf are still the most common performance limiters
  - E.g. 8kB window, 87 msec ping time = 753 kbps
  - E.g. 64kB window, 14 msec rtt = 37 Mbps

Dijkstra, SC2002

67

## Maximum TCP/IP Data Rate With 64KB window



Dijkstra, SC2002

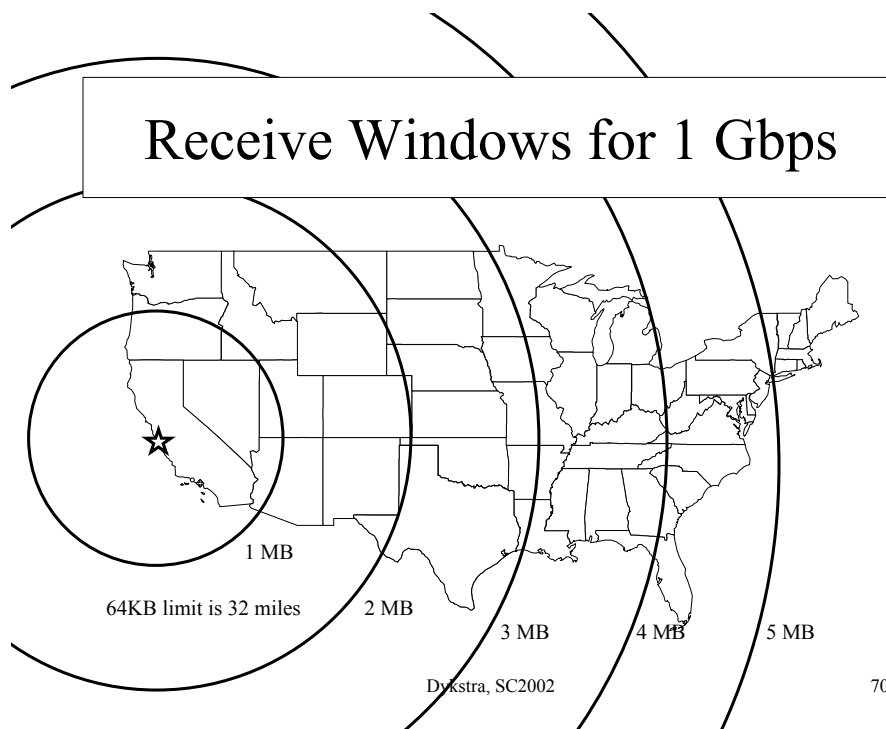
68

## Bandwidth\*Delay Product and TCP

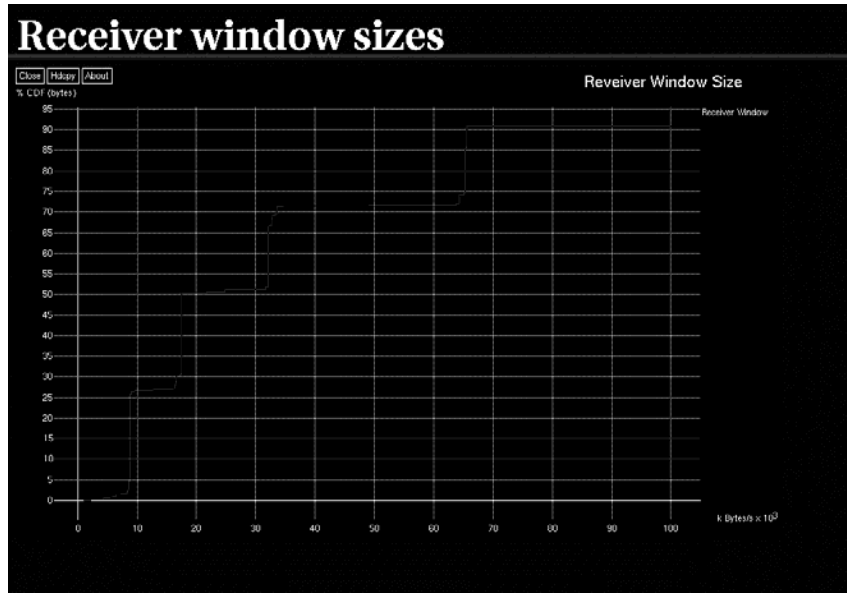
- TCP needs a **receive window** (rwin) equal to or greater than the  $BW * Delay$  product to achieve maximum throughput
- TCP needs **sender side socket buffers** of  $2 * BW * Delay$  to recover from errors
- You need to send about  $3 * BW * Delay$  bytes for TCP to reach maximum speed

Dijkstra, SC2002

69



70



Dykstra, SC2002

71

## Observed Receiver Window Sizes

- ATM traffic from the Pittsburgh Gigapop
- 50% have windows < 20 KB
  - These are obsolete systems!
- 20% have 64 KB windows
  - Limited to ~ 8 Mbps coast-to-coast
- ~9% are assumed to be using window scale

M. Mathis, PSC

Dykstra, SC2002


72

# System Tuning

Interfaces, routes, buffers, etc.

## Things You Can Do



- Throw out your low speed interfaces and networks! 
- Make sure routes and DNS report high speed interfaces
- Don't over-utilize your links (<50%)
- Use routers sparingly, host routers not at all  
`routed -q`

# Things You Can Do



- Make sure your HPC apps offer sufficient receive windows and use sufficient send buffers
  - But don't run your system out of memory
  - Find out the rtt with ping, compute BDP
  - Can tune system wide, by application, or automatically
- Check your TCP for high performance features
- Look for sources of loss
  - Watch out for duplex problems (late collisions?)

Dijkstra, SC2002

75

## System Tuning: Linux 2.4

```
/etc/sysctl.conf
# Increase max socketbuffer sizes, actual = 2x these values
net.core.rmem_max = 1048576
net.core.wmem_max = 1048576
# Increase min, default, auto buffer sizes
net.ipv4.tcp_rmem = 4096      349520  699040
net.ipv4.tcp_wmem = 4096      65536   524288
# Remove ICMP reply rate limits (now icmp_ratemask)
net.ipv4.icmp_echoreply_rate = 0
net.ipv4.icmp_destunreach_rate = 0
# Misc performance features (defaults)
net.ipv4.ip_no_pmtu_disc = 0
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_ecn = 0
```

Dijkstra, SC2002

76

# System Tuning: FreeBSD

```
# FreeBSD 3.4 defaults are 524288 max, 16384 default
/sbin/sysctl -w kern.ipc.maxsockbuf=1048576
/sbin/sysctl -w net.inet.tcp.sendspace=32768
/sbin/sysctl -w net.inet.tcp.recvspace=32768
```

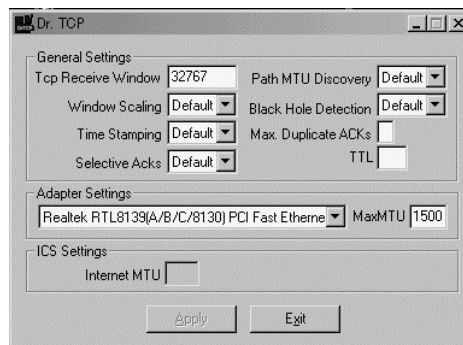
Dykstra, SC2002

77

## Dr. TCP

A TCP Stack Tuner for Windows

<http://www.dslreports.com/front/drtcp.html>



- See also, <http://cable-dsl.home.att.net/> for registry edits, etc.
- Beware that modem utilities such as DunTweak can reduce performance on high speed nets

Dykstra, SC2002

78

# Ethernet Duplex Problems

## An Internet Epidemic!

- Ethernet “auto-negotiation” can select the speed and duplex of a connected pair
- If only one end is doing it:
  - It can get the speed right
  - but it will **assume half-duplex**
- Mismatch loss only shows up under load
  - Can’t see it with ping

Dijkstra, SC2002

79

# Tuning FTP

- Many FTP’s allow the user to set buffer sizes
- The commands are different everywhere!
- See <http://dast.nlanr.net/Projects/FTP.html> for several sets of commands

Dijkstra, SC2002

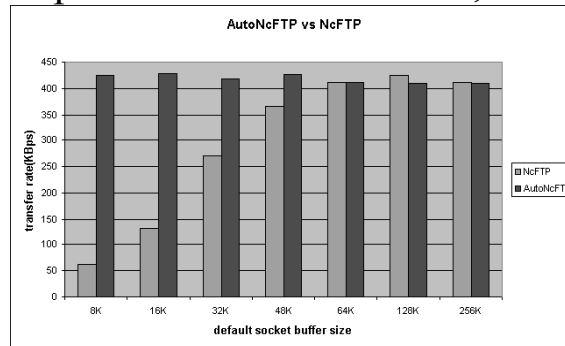
80



# Autobuf – An Auto-tuning FTP

<http://dast.nlanr.net/Projects/Autobuf/>

- Measures the spread of a burst of ICMP Echo packets to estimate BDP, sets bufs



Dykstra, SC2002

81

## Good Tuning References

- Users Guide to TCP Windows  
[www.ncsa.uiuc.edu/People/vwelch/net\\_perf/tcp\\_windows.html](http://www.ncsa.uiuc.edu/People/vwelch/net_perf/tcp_windows.html)
- TCP Tuning Guide  
[www-didc.lbl.gov/TCP-tuning/](http://www-didc.lbl.gov/TCP-tuning/)
- Enabling High Performance Data Transfers on Hosts  
[www.psc.edu/networking/perf\\_tune.html](http://www.psc.edu/networking/perf_tune.html)

Dykstra, SC2002

82

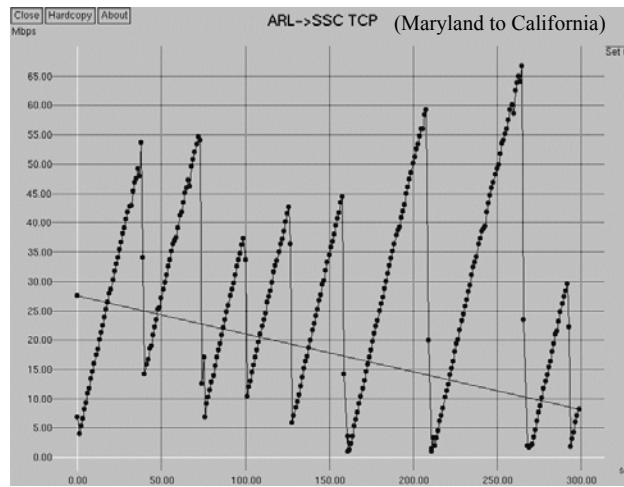
# TCP

The Internet's transport

## Important Points About TCP

- TCP is *adaptive*
- It is *constantly* trying to go *faster*
- It *slows down* when it detects a *loss*
- *How much* it sends is controlled by *windows*
- *When* it sends is controlled by *received ACK's* (or timeouts)

## TCP Throughput vs. Time



Dykstra, SC2002

85

## AIMD

- Additive Increase, Multiplicative Decrease
  - of  $c_{win}$ , the congestion window
- The core of TCP's congestion avoidance phase, or "steady state"
  - Standard increase = +1.0 MSS per loss free rtt
  - Standard decrease =  $\times 0.5$  (i.e. halve  $c_{win}$  on loss)
- Avoids congestion collapse
- Promotes fairness among flows

Dykstra, SC2002

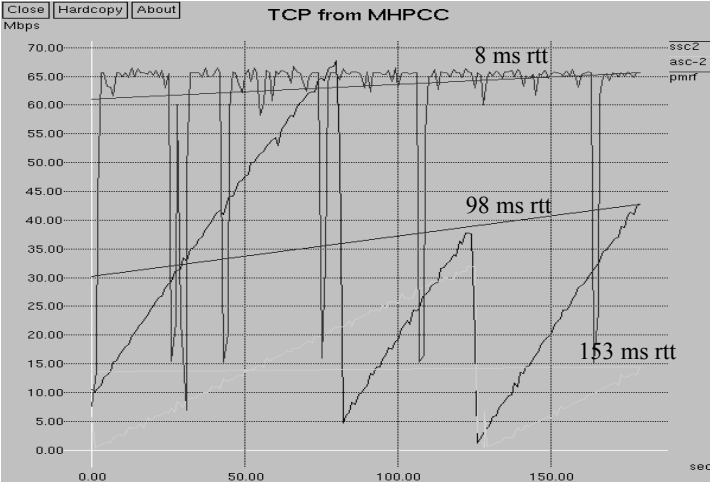
86

# Iperf : TCP California to Ohio

```
damp-ssc2$ iperf -c damp-asc2 -p56117 -w750k -t20 -i1 -fm
-----
Client connecting to damp-asc2, TCP port 56117
TCP window size: 1.5 MByte (WARNING: requested 0.7 MByte)
-----
[ 3] local 192.168.25.74 port 35857 connected with 192.168.244.240 port 56117
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0- 1.0 sec    1.7 MBytes  14.2 Mb/s/sec
[ 3] 1.0- 2.3 sec    1.5 MBytes   9.5 Mb/s/sec
[ 3] 2.3- 3.2 sec    1.2 MBytes  11.2 Mb/s/sec
[ 3] 3.2- 4.1 sec    1.2 MBytes  12.2 Mb/s/sec
[ 3] 4.1- 5.1 sec    1.6 MBytes  12.5 Mb/s/sec
[ 3] 5.1- 6.1 sec    1.6 MBytes  13.6 Mb/s/sec
[ 3] 6.1- 7.0 sec    1.6 MBytes  14.6 Mb/s/sec
[ 3] 7.0- 8.2 sec    2.0 MBytes  14.7 Mb/s/sec
[ 3] 8.2- 9.1 sec    1.6 MBytes  15.4 Mb/s/sec
[ 3] 9.1-10.1 sec    2.0 MBytes  16.7 Mb/s/sec
[ 3] 10.1-11.1 sec   2.0 MBytes  17.0 Mb/s/sec
[ 3] 11.1-12.0 sec   2.0 MBytes  18.5 Mb/s/sec
[ 3] 12.0-13.1 sec   2.4 MBytes  18.8 Mb/s/sec
[ 3] 13.1-14.1 sec   2.4 MBytes  19.1 Mb/s/sec
[ 3] 14.1-15.1 sec   2.4 MBytes  20.8 Mb/s/sec
[ 3] 15.1-16.1 sec   2.4 MBytes  20.6 Mb/s/sec
[ 3] 16.1-17.0 sec   2.4 MBytes  22.0 Mb/s/sec
[ 3] 17.0-18.1 sec   2.8 MBytes  22.2 Mb/s/sec
[ 3] 18.1-19.1 sec   1.6 MBytes  13.6 Mb/s/sec
[ 3] 19.1-20.1 sec   1.6 MBytes  12.3 Mb/s/sec
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-20.6 sec  38.2 MBytes  15.5 Mb/s/sec
```

82 msec rtt

# TCP Examples from Maui HI



## TCP Acceleration ( $MSS/rtt^2$ )

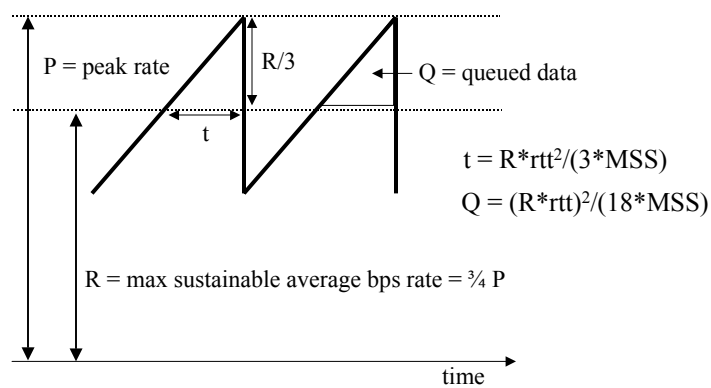
(Congestion avoidance rate increase,  $MSS = 1448$ )

rtt (msec)	Mbps/s	0-100Mbps (sec)
5	463	0.216
10	116	0.864
20	29	3.45
50	4.6	21.6
100	1.16	86.4
200	0.29	345

Dijkstra, SC2002

89

## TCP Average Rate



Dijkstra, SC2002

90

## Observations

- Low latency really helps performance!
- Queued data is proportional to the square of the bandwidth delay product ( $BDP^2$ )
  - 100 KB BDP needs ~400 KB queue
  - 1 MB BDP needs ~40 MB queue!
- Queue duration is proportional to the square of the round trip time ( $rtt^2$ )
- Jumbo frames help **reduce** the size and duration of queues by increasing the slope (TCP acceleration)

Dijkstra, SC2002

91

## Observations

- For high BDP's, average rate will be at best three fourths of the tight link rate
- To do better requires throttling the sender in a way that doesn't halve *cwin* at peaks, e.g.
  - carefully setting *rwin*
  - pacing the sender
  - TCPW (sets *cwin* based on estimated BW)

Dijkstra, SC2002

92

# TCP Throughput

[www.psc.edu/networking/papers/model\\_abstract.html](http://www.psc.edu/networking/papers/model_abstract.html)

*Once recv window size and available bandwidth aren't the limit*

$$\text{Rate} = \frac{\sim 0.7 * \text{Max Segment Size (MSS)}}{\text{Round Trip Time (latency)} \sqrt{\text{pkt\_loss}}}$$

M. Mathis, et al.

- Double the MTU, double the throughput
- Halve the latency, double the throughput
  - shortest path matters
- Halve the loss rate, 40% higher throughput

Dijkstra, SC2002

93

## Max Segment Size (MSS)

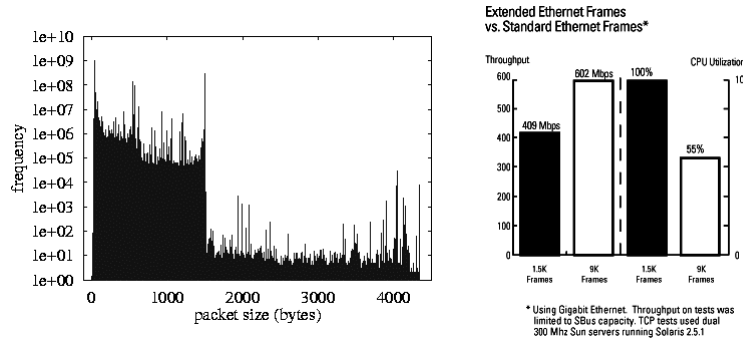
$$\text{rate} = 0.7 * \text{MSS} / (\text{rtt} * \sqrt{p})$$

- MSS = MTU – packet headers
- Common MTU's
  - 576 IPv4 default
  - 1500 ethernet, IPv6 default
  - ~9000 GigE Jumbo Frame, CLIP ATM
  - 64k max ATM AAL5 frame
- Jumbo frame => ~6x throughput increase

Dijkstra, SC2002

94

# Packet Size (MTU) Issues



<http://sd.wareonearth.com/~phil/jumbo.html>

“New York to Los Angeles. Round Trip Time (rtt) is about 40 msec, and let's say packet loss is 0.1% (0.001). With an MSS of 1460 bytes, TCP throughput will have an upper bound of about 6.5 Mbps! And no, that is not a window size limitation, but rather one based on TCP's ability to detect and recover from congestion (loss). With 9000 byte frames, TCP throughput could reach about 40 Mbps.”

Dijkstra, SC2002

95

## Path MTU

- Maximum Transmission Unit (MTU)
  - Largest packet that can be sent as a unit
- Path MTU (PMTU)
  - min MTU of all hops in a path
- Hosts can do Path MTU Discovery to find it
  - Depends on ICMP replies
- Without PMTU Discovery hosts should assume PMTU is only 576 bytes
  - Some hosts falsely assume 1500!

Dijkstra, SC2002

96



## Things You Can Do



- Use only large MTU interfaces/routers/links
  - Gigabit Ethernet with **Jumbo Frames** (9000)
  - ATM CLIP (9180)
  - Packet over SONET (POS) (4470, 9000+)
- Never reduce the MTU (or bandwidth) on the path between each/every host and the WAN
- Make sure your TCP uses Path MTU Discovery

Dijkstra, SC2002

97

## Round Trip Time (RTT)

$$\text{rate} = 0.7 * \text{MSS} / (\text{rtt} * \text{sqrt}(p))$$

- If we could halve the delay we could double throughput!
- Most delay is caused by speed of light in fiber (~200 km/msec)
- “Scenic routing” and fiber paths raise the minimum
- Congestion (queueing) adds delay

Dijkstra, SC2002

98

## Packet Loss (p)

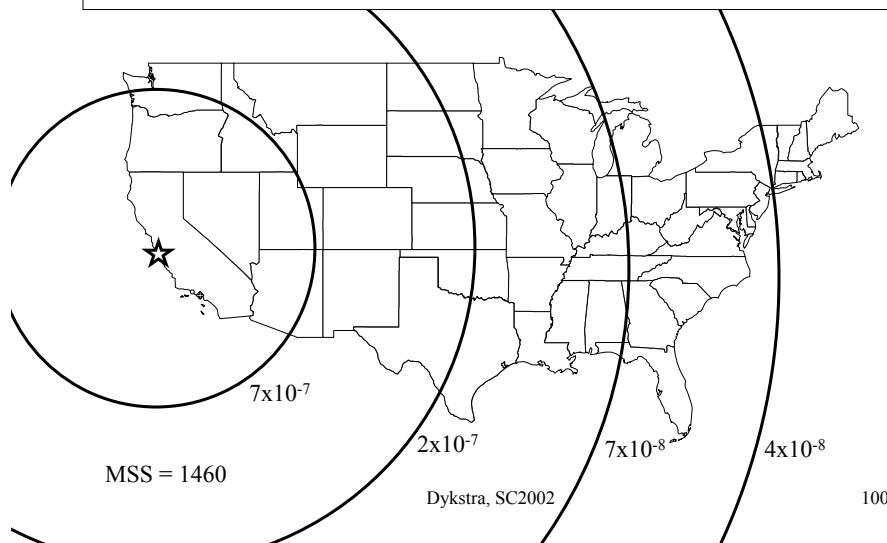
$$\text{rate} = 0.7 * \text{MSS} / (\text{rtt} * \sqrt{p})$$

- ***Loss dominates throughput !***
- At least 6 orders of magnitude observed on the Internet
- 100 Mbps throughput requires  $O(10^{-6})$
- 1 Gbps throughput requires  $O(10^{-8})$

Dykstra, SC2002

99

### Loss Limits for 1 Gbps



## Specifying Loss

- TCP loss limits for 1 Gbps across country are  $O(10^{-8})$ , i.e. 0.000001% packet loss
  - About 1 “ping” packet every three years
  - *Systems like AMP would never show loss*
  - Try to get  $10^{-8}$  in writing from a provider!
  - Most providers won't guarantee  $< 0.01\%$

Dijkstra, SC2002

101

## Specifying Throughput

- Require the provider to demonstrate TCP throughput
  - New DREN RFP requires  $\frac{1}{2}$  line rate TCP flow sustained for 10 minutes cross country (e.g. ~300 Mbps on OC12)
- A low loss requirement comes with this!

Dijkstra, SC2002

102

## Concerns About Bit Errors

- Bit Error Rate (BER) specs for networking interfaces/circuits may not be low enough
  - E.g.  $10^{-12}$  BER  $\Rightarrow$   $10^{-8}$  packet ER (1500 bytes)
  - 10 hops  $\Rightarrow$   $10^{-7}$  packet drop rate
- CRC32 and checksums may be too weak
- Error detection in the NIC may miss some
  - E.g. DMA errors (Partridge et al.)

Dijkstra, SC2002

103

## More About TCP

Some details

## TCP Keeps Evolving

- TCP, RFC 793, Sep 1981
- Reno, BSD, 1990
- Path MTU Discovery, RFC 1191, Nov 1990
- Window Scale, PAWS, RFC 1323, May 1992
- SACK, RFC 2018, Oct 1996
- NewReno, RFC 2581, April 1999
- D-SACK, RFC 2883, July 2000
- More on the way!

Dijkstra, SC2002

105

## TCP Reno

- Most modern TCP's are "Reno" based
- Reno defined (refined) four key mechanisms
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit
  - Fast Recovery
- NewReno refined fast retransmit/recovery when partial acknowledgements are available

Dijkstra, SC2002

106

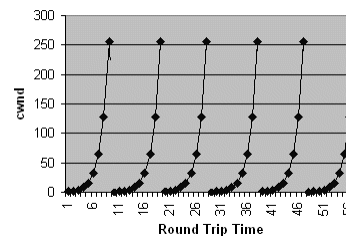
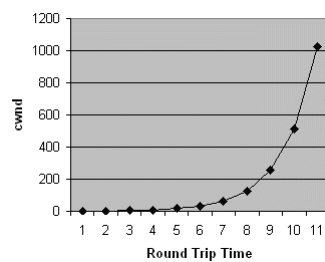
# TCP Congestion Window (cwin)

- Congestion window controls startup and limits throughput in the face of loss
- cwin gets larger after every new ACK
- cwin get smaller when loss is detected
- cwin amounts to TCP's estimate of the available bandwidth at any given time

Dijkstra, SC2002

107

## Cwin During Slowstart

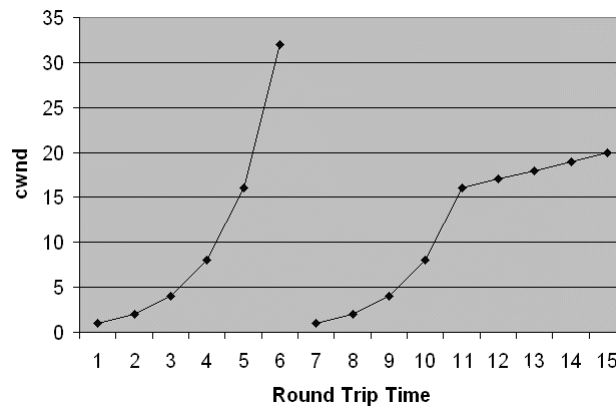


- cwin increased by one for every new ACK
- cwin doubles every round trip time (exponential)
- cwin is reset to zero after a loss

Dijkstra, SC2002

108

## Slowstart and Congestion Avoidance Together



Dijkstra, SC2002

109

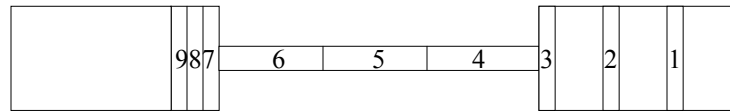
## Delayed ACKs

- TCP receivers send ACK's:
  - after every second segment
  - after a delayed ACK timeout
  - on every segment after a loss (missing segment)
- A new segment sets the delayed ACK timer
  - Typically 0-200 msec
- A second segment (or timeout) triggers an ACK and clears the delayed ACK timer

Dijkstra, SC2002

110

## ACK Clocking



- A queue forms in front of a slower speed link
- The slower link causes packets to spread
- The spread packets result in spread ACK's
- The spread ACK's end up clocking the source packets at the slower link rate

Dijkstra, SC2002

111

## Detecting Loss

- Packets get discarded when queues are full (or nearly full)
- Duplicate ACK's get sent after missing or out of order packets
- Most TCP's retransmit after the third duplicate ACK ("triple duplicate ACK")
  - Windows XP now uses 2<sup>nd</sup> dup ACK

Dijkstra, SC2002

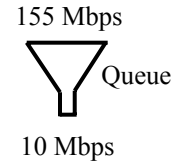
112



# Random Early Detection (RED)

RFC 2309, Apr 1998

- Discards arriving packets as a function of queue length
- Gives TCP better congestion indications (drops)
- Avoids “Global Synchronization”
- Increases total number of drops
- Increases link utilization
- Many variations (weighted, classed, etc.)



Dijkstra, SC2002

113

## SACK TCP

### Selective Acknowledgement

- SACK specifies exactly which bytes were missed
- D-SACK can specify which bytes were duplicated
- Better measures the “right edge” of the congestion window (i.e. most recently received data)
- Can do a **very** good job keeping your queues full
  - Which causes latencies to go way up
- Without RED, will cause global sync faster
- Win98, Win2k, Linux have SACK

Dijkstra, SC2002

114

## Things You Can Do



- Consider using RED on your routers before wide scale deployment of SACK TCP
- SACK won't care very much but your old TCP's will thank you
- Consider a priority class of service for interactive traffic?



Dijkstra, SC2002

115

## Advanced Debugging

TCP Traces and Testrig

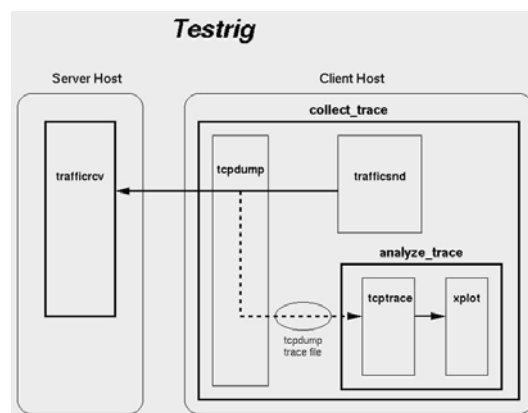
## TCP/IP Analysis Tools

- tcpdump
  - [www.tcpdump.org](http://www.tcpdump.org)
- ethereal - GUI tcpdump (protocol analyzer)
  - [www.ethereal.com](http://www.ethereal.com)
- tcptrace – stats/graphs of tcpdump data
  - [www.tcptrace.org](http://www.tcptrace.org)
- testrig – tcpdump, tcptrace, xplot, etc.
  - [www.ncne.nlanr.net/research/tcp/testrig/](http://www.ncne.nlanr.net/research/tcp/testrig/)

Dykstra, SC2002

117

## “A Preconfigured TCP Test Rig”



Dykstra, SC2002

118

```

TCP connection 1:
  host a:      sd.wareonearth.com:1095
  host b:      amp2.sd.wareonearth.com:56117
  complete conn: yes
  first packet: Sun Apr 23 23:35:29.645263 2000
  last packet:  Sun Apr 23 23:35:41.108465 2000
  elapsed time: 0:00:11.463202
  total packets: 107825
  filename:     trace.0.20000423233526

a->b:
  total packets:      72032
  ack pkts sent:      72031
  pure acks sent:      2
  unique bytes sent:  104282744
  actual data pkts:    72029
  actual data bytes:   104282744
  rexmt data pkts:     0
  rexmt data bytes:    0
  outoforder pkts:     0
  pushed data pkts:    72029
  SYN/FIN pkts sent:  1/1
  req 1323 ws/ts:      Y/Y
  adv wind scale:      0
  req sack:            Y
  sacks sent:          0
  mss requested:       1460 bytes
  max segm size:       1448 bytes
  min segm size:       448 bytes
  avg segm size:       1447 bytes
  max win adv:         32120 bytes
  min win adv:         32120 bytes
  zero win adv:        0 times
  avg win adv:         32120 bytes
  initial window:      2896 bytes
  initial window:      2 pkts
  ttl stream length:   104857600 bytes
  missed data:         974856 bytes
  truncated data:      101833758 bytes
  truncated packets:   72029 pkts
  data xmit time:      11.461 secs
  idletime max:        372.0 ms
  throughput:          9097174 Bps

b->a:
  total packets:      35793
  ack pkts sent:      35793
  pure acks sent:      35791
  unique bytes sent:    0
  actual data pkts:     0
  actual data bytes:    0
  rexmt data pkts:     0
  rexmt data bytes:    0
  outoforder pkts:     0
  pushed data pkts:    0
  SYN/FIN pkts sent:   1/1
  req 1323 ws/ts:      Y/Y
  adv wind scale:      4
  req sack:            N
  sacks sent:          0
  mss requested:       1460 bytes
  max segm size:       0 bytes
  min segm size:       0 bytes
  avg segm size:       0 bytes
  max win adv:         750064 bytes
  min win adv:         65535 bytes
  zero win adv:        0 times
  avg win adv:         30076 bytes
  initial window:      0 bytes
  initial window:      0 pkts
  ttl stream length:   0 bytes
  missed data:         0 bytes
  truncated data:      0 bytes
  truncated packets:   0 pkts
  data xmit time:      0.000 secs
  idletime max:        246.8 ms
  throughput:          0 Bps

```

## tcptrace -l

Dykstra, SC2002

119

## TCP Connection Establishment

- Three-way handshake
  - SYN, SYN+ACK, ACK
- Use tcpdump, look for performance features
  - window sizes, window scale, timestamps, MSS, SackOK, Don't-Fragment (DF)

Dykstra, SC2002

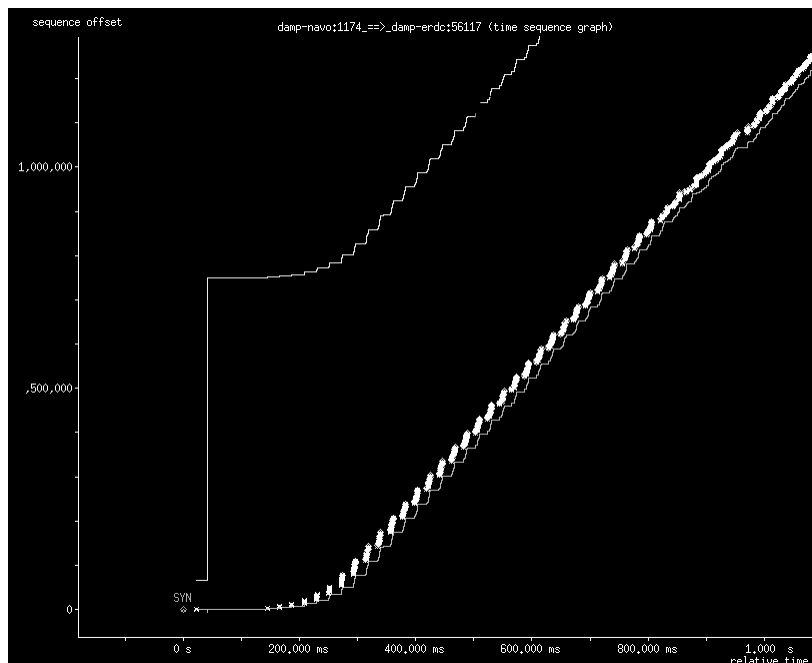
120

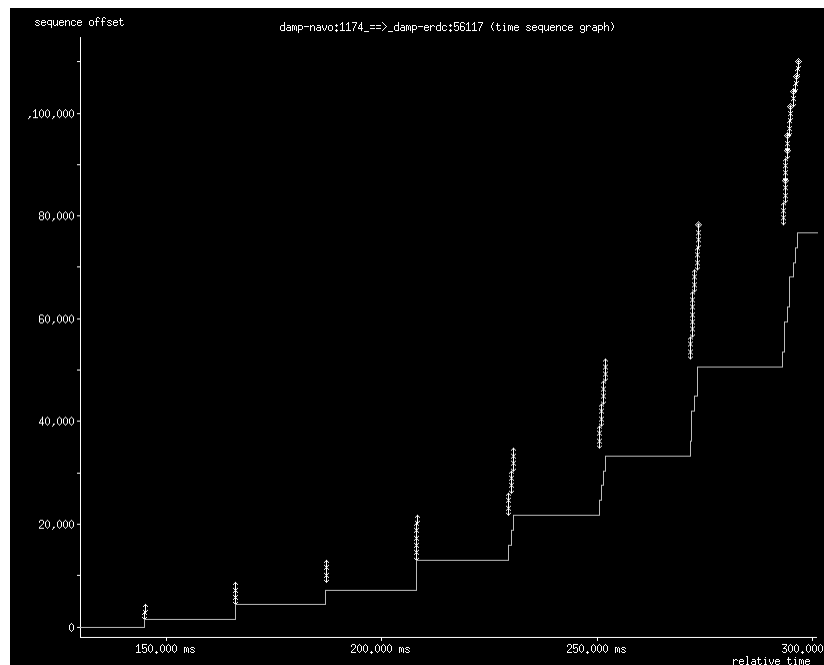
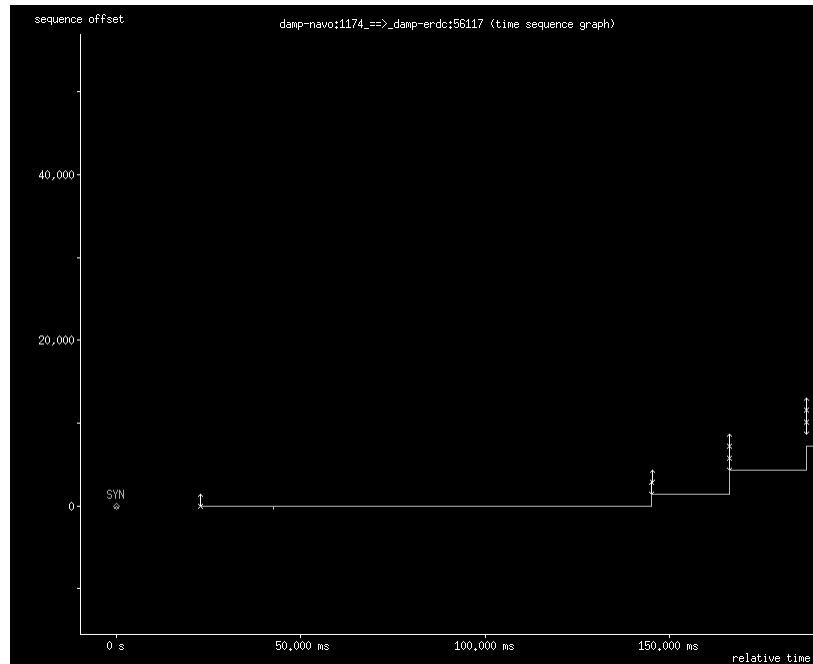
# Tcpdump of TCP Handshake

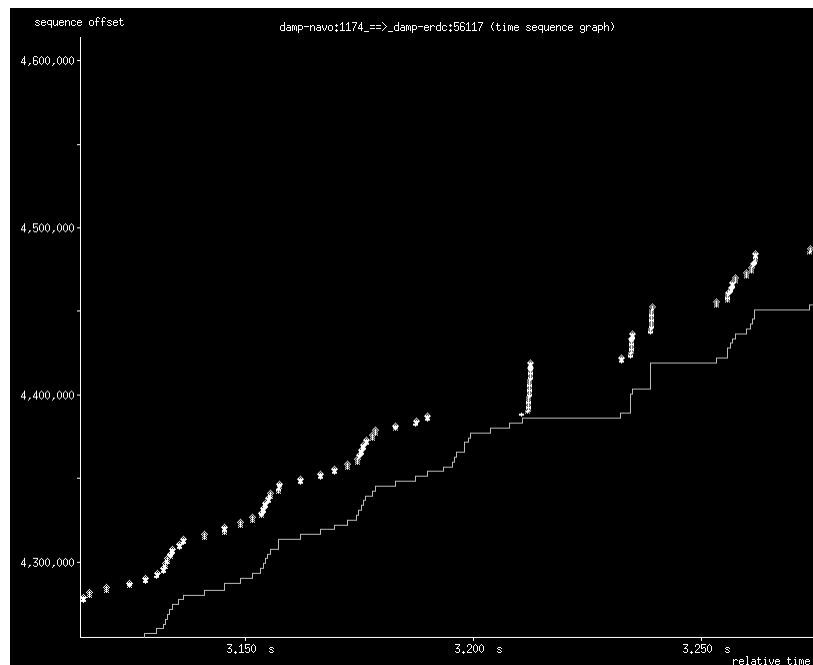
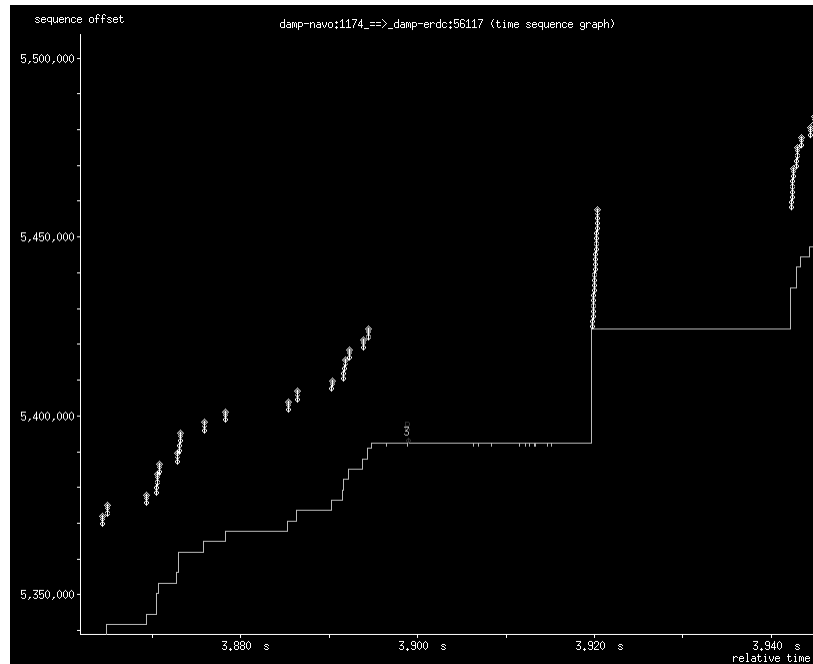
```
16:08:33.674226 wcisd.hpc.mil.40874 > damp-nrl.56117:  
  S 488615735:488615735(0) win 5840  
  <mss 1460,sackOK,timestamp 263520790 0,nop,wscale 0> (DF)  
  
16:08:33.734045 damp-nrl.56117 > wcisd.hpc.mil.40874:  
  S 490305274:490305274(0) ack 488615736 win 5792  
  <mss 1460,sackOK,timestamp 364570771 263520790,nop,wscale 5> (DF)  
  
16:08:33.734103 wcisd.hpc.mil.40874 > damp-nrl.56117:  
  . ack 1 win 5840  
  <nop,nop,timestamp 263520796 364570771> (DF)
```

Dijkstra, SC2002

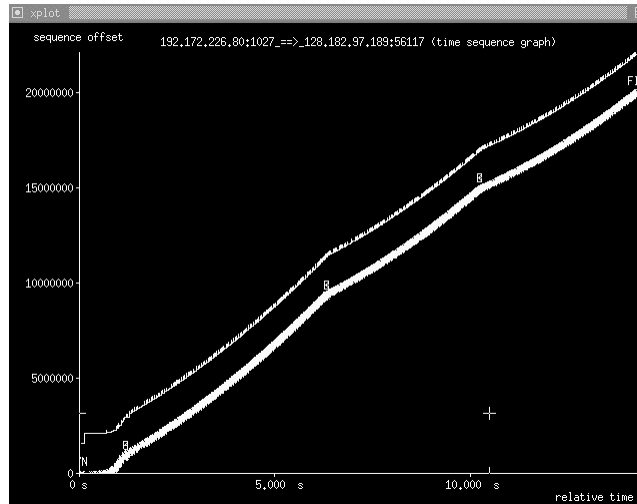
121







## Normal TCP Scallop

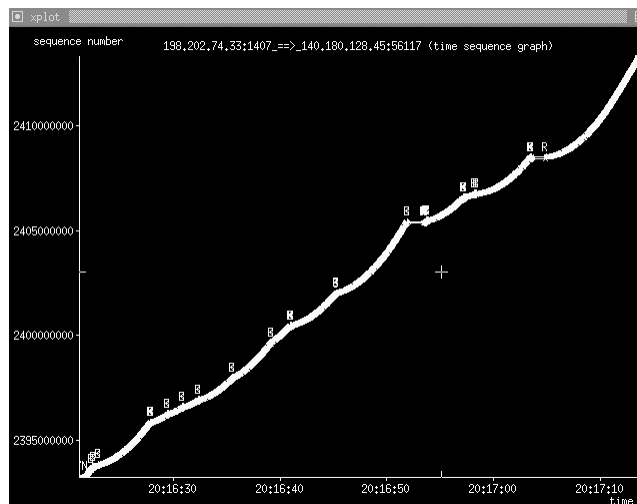


NLANR  
NCNE

Dykstra, SC2002

127

## A Little More Loss



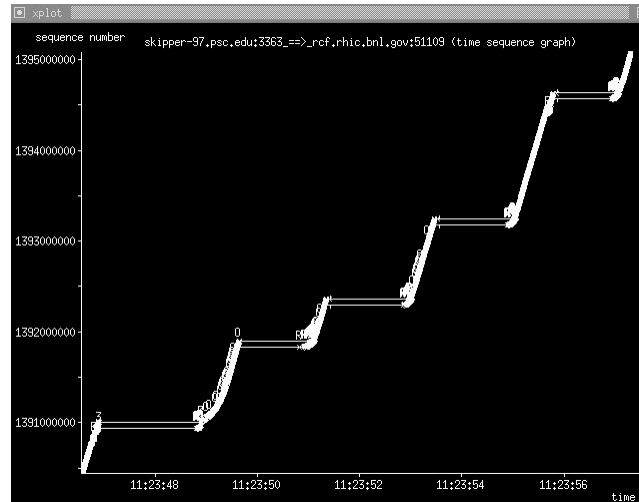
NLANR  
NCNE

Dykstra, SC2002

128



# Excessive Timeouts

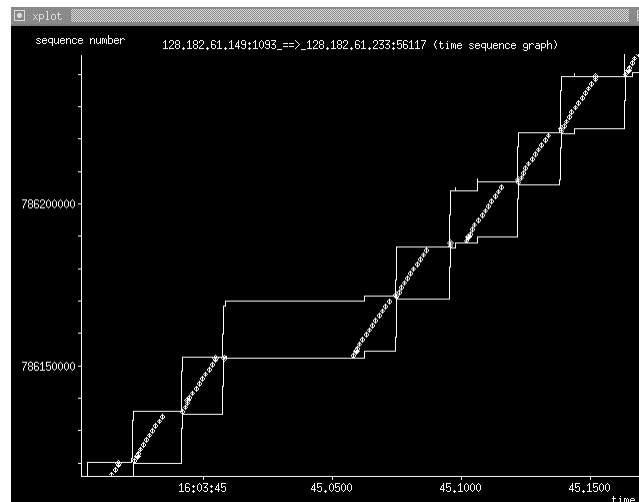


NLANR  
NCNE

Dijkstra, SC2002

129

# Bad Window Behavior

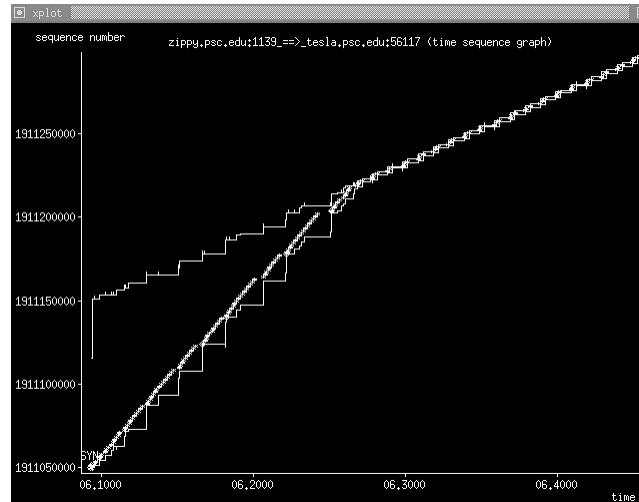


NLANR  
NCNE

Dijkstra, SC2002

130

## Receiving Host/App Too Slow



NLANR  
NCNE

Dykstra, SC2002

131

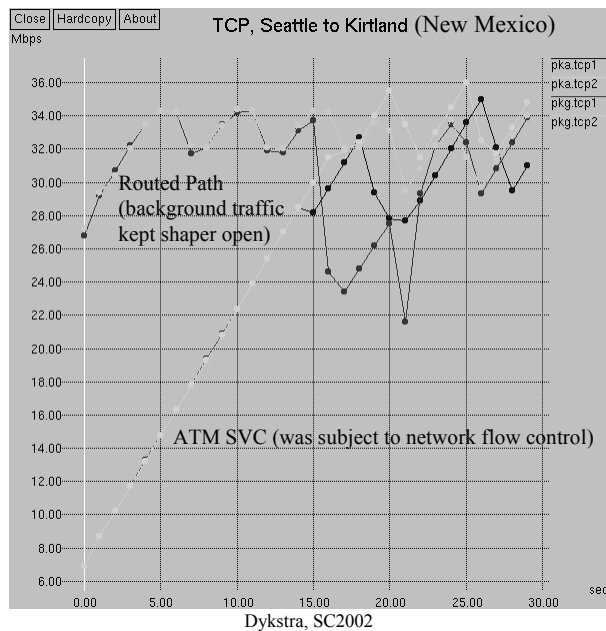
## Traffic Management

*Beware of network flow control*

- ATM Forum TM 4.1
  - For UBR and ABR flows (Unspecified and Available Bit Rate)
  - Blocks cells from entering the WAN
  - “Opens up” over time if BW is available
- Can destroy the performance of short flows
- Similar TM mechanisms are being considered for MPLS!

Dykstra, SC2002

132



133

## nuttcp: One Second UDP Test

```
damp-kirt$ nuttcp -t -T1 -u -R30m damp-ssc2
nuttcp-t: v3.1.9: socket
nuttcp-t: buflen=8192, nstream=1, port=5001 udp -> damp-ssc2
nuttcp-t: time limit = 1 second
nuttcp-t: rate limit = 30.000 Mb/s
nuttcp-t: send window size = 65535, receive window size = 65535
nuttcp-t: 3.578 MB in 1.00 real seconds = 3662.08 KB/sec = 29.9997 Mb/s
nuttcp-t: 464 I/O calls, msec/call = 2.21, calls/sec = 463.76
nuttcp-t: 0.5user 0.4sys 0:01real 100% 0i+0d 0maxrss 0+0pf 0+0csw

nuttcp-r: v3.1.9: socket
nuttcp-r: buflen=8192, nstream=1, port=5001 udp
nuttcp-r: send window size = 65535, receive window size = 65535
nuttcp-r: 0.609 MB in 2.02 real seconds = 309.21 KB/sec = 2.5331 Mb/s
nuttcp-r: 82.98% data loss
nuttcp-r: 80 I/O calls, msec/call = 25.83, calls/sec = 39.64
nuttcp-r: 0.0user 0.0sys 0:02real 0% 0i+0d 0maxrss 0+0pf 0+0csw
```

Dijkstra, SC2002

134

# Web100

[www.web100.org](http://www.web100.org)

- Set out to make 100 Mbps TCP common
- “TCP knows what’s wrong with the network”
  - Mostly on the sender side
- Instruments the TCP stack for diagnostics
- Enhanced TCP MIB (IETF Draft)
- Linux 2.4 kernel patches + library and tools
- /proc/web100 file system
  - e.g. /proc/web100/1010/{read,spec,spec-ascii,test,tune}

Dijkstra, SC2002

135

## Web100 – Connection Selection

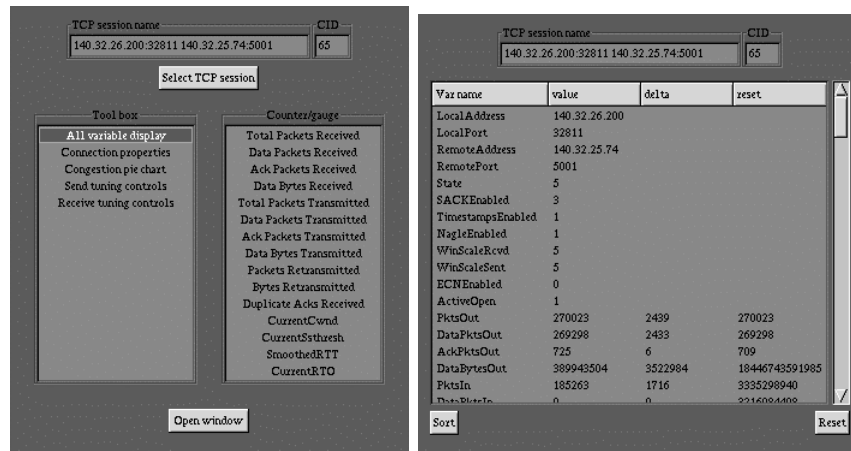
cmdline	pid	local add	local port	remote add	remote port	cid	state
	0	127.0.0.1	9753	127.0.0.1	32814	69	---
	0	127.0.0.1	32814	127.0.0.1	9753	68	---
nuttcp	2338	140.32.26.200	32811	140.32.25.74	5001	65	ESTBLSH
nuttcp	2338	140.32.26.200	32810	140.32.25.74	5000	64	ESTBLSH
nuttcp	2338	140.32.26.200	32810	140.32.25.74	5000	64	ESTBLSH
	0	127.0.0.1	6010	127.0.0.1	32796	49	ESTBLSH
gutil	2173	127.0.0.1	32796	127.0.0.1	6010	48	ESTBLSH
	0	192.168.0.1	22	192.168.0.100	60642	47	---
	0	140.32.26.200	1090	198.253.246.7	4635	33	ESTBLSH
	0	140.32.26.200	1090	63.196.71.245	1834	32	ESTBLSH
	0	140.32.26.200	1090	198.97.151.50	1034	31	ESTBLSH
	0	140.32.26.200	1090	140.32.26.61	34694	30	ESTBLSH
	0	140.32.26.200	1090	132.250.100.154	38024	29	ESTBLSH
	0	140.32.26.200	1090	199.165.80.6	39728	24	ESTBLSH
	0	140.32.26.200	1090	204.222.178.138	1067	23	ESTBLSH
	0	140.32.26.200	1090	140.31.150.2	1041	20	ESTBLSH

Sort entries by:

Dijkstra, SC2002

136

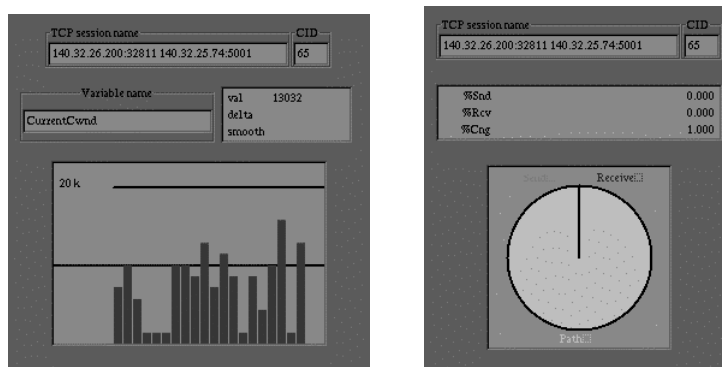
## Web100 - Tool/Variable Selection



Dijkstra, SC2002

137

## Web100 – Variable Display, Triage Chart



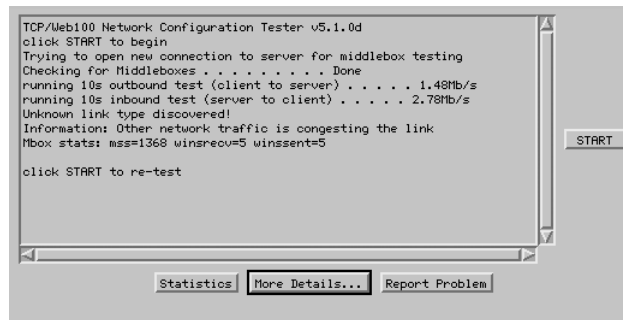
See also [www.net100.org](http://www.net100.org) for more work based on Web100

Dijkstra, SC2002

138

# ANL Network Tester

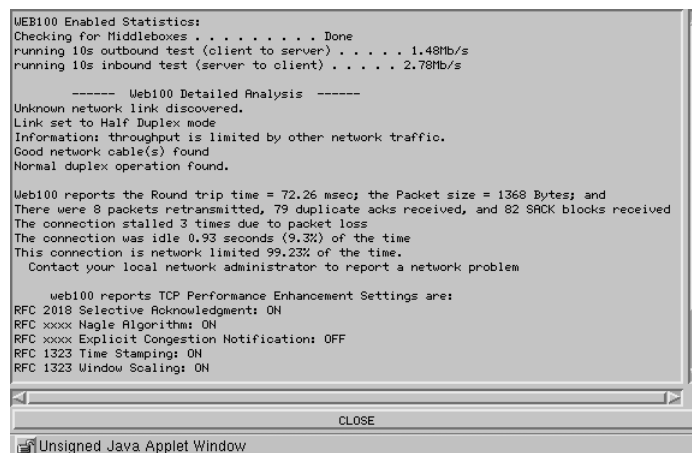
<http://miranda.ctd.anl.gov:7123/>



Dijkstra, SC2002

139

## ANL Tester - Statistics



Dijkstra, SC2002

140

# Iperf with Web100, Clean Link

```
wcisd$ iperf-web100 -e -w400k -p56117 -c damp-wcisd
-----
Client connecting to damp-wcisd, TCP port 56117
TCP window size: 800 KByte (WARNING: requested 400 KByte)
-----
[ 3] local 192.168.26.200 port 33185 connected with 192.168.26.61 port 56117
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  113 MBytes  94.1 Mbits/sec
----- Web100 Analysis -----

100 Mbps FastEthernet link found
Good network cable(s) found
Duplex mismatch condition NOT found
Link configured for Full Duplex operation
Information: This link is congested with traffic

Web100 reports the Round trip time = 14.0 msec; the Packet size = 1448 Bytes; and
There were 1 packets retransmitted, 0 duplicate acks received, and 0 SACK blocks received
This connection is network limited 99.99% of the time.
Contact your local network administrator to report a network problem

Web100 reports the Tweakable Settings are:
RFC-1323 Time Stamping: On
RFC-1323 Window Scaling Option: On
RFC-2018 Selective Acknowledgment (SACK): On
```

Dijkstra, SC2002

141

# Iperf with Web100, Lossy Link

```
wcisd$ iperf-web100 -e -w400k -p56117 -c damp-ssc2
-----
Client connecting to damp-ssc2, TCP port 56117
TCP window size: 800 KByte (WARNING: requested 400 KByte)
-----
[ 3] local 192.168.26.200 port 33198 connected with 192.168.25.74 port 56117
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.2 sec  35.0 MBytes  28.9 Mbits/sec
----- Web100 Analysis -----

Unknown link type found
Good network cable(s) found
Warning: Duplex mismatch condition exists: Host HD and Switch FD
Information: link configured for Half Duplex operation
No congestion found on this link

Web100 reports the Round trip time = 2.0 msec; the Packet size = 1448 Bytes; and
There were 617 packets retransmitted, 4072 duplicate acks received, and 4370 SACK blocks received
The connection stalled 1 times due to packet loss
The connection was idle for 0.21 seconds (2.06%) of the time
This connection is network limited 99.99% of the time.
Contact your local network administrator to report a network problem
```

Dijkstra, SC2002

142

# Going Faster

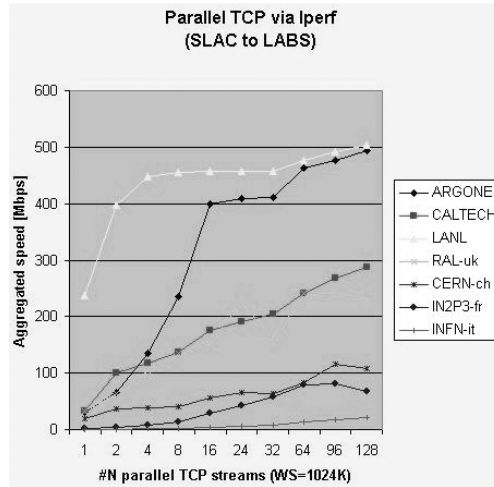
Cheating Today, Improving TCP  
Tomorrow

## Parallel TCP Streams

- PSocketS (Parallel Sockets library)
  - <http://citeseer.nj.nec.com/386275.html>
- GridFTP – enhanced parallel wu-ftpd
  - <http://www.globus.org/datagrid/gridftp.html>
- bbFTP – parallel ‘ftp’ uses ssh or GSI
  - <http://doc.in2p3.fr/bbftp/>
- MulTCP – a TCP that acts like N TCP’s
  - UCL



# Parallel TCP Stream Performance

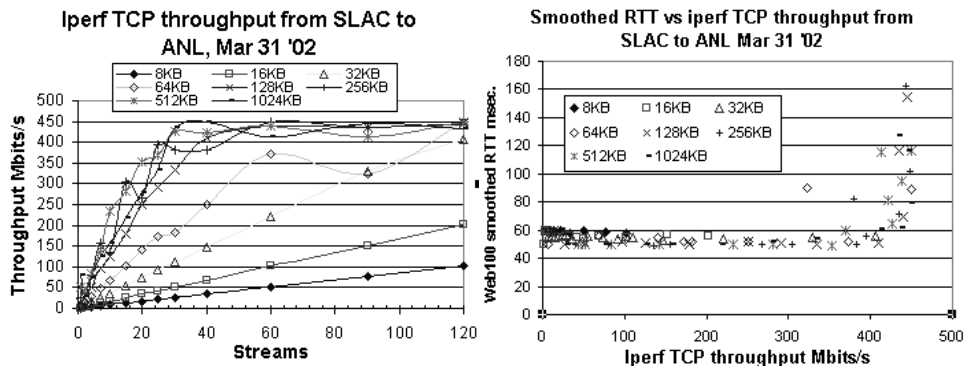


Dykstra, SC2002

Les Cottrell, SLAC

145

## Parallel TCP Streams Throughput and RTT by Window Size



Les Cottrell, SLAC

Dykstra, SC2002

146

## Tsunami - Use UDP Instead

<http://www.anml.iu.edu/>

- UDP data stream, constant rate
- TCP control stream to request retransmissions
- Transferred 1 TB of data at ~1 Gbps over a 12000 km “light path” (Vancouver to Geneva), Sep 2002
- Was created because TCP over that path was getting only 10’s to 100’s of Mbps!

Dijkstra, SC2002

147

## The Future of TCP/IP

- Different retransmit/recovery schemes
  - TCP Tahoe, Vegas, Peach, Westwood, ...
- Pacing - removing burstiness by spreading the packets over a round trip time (BLUE)
- Rate-halving to recover ACK clocking more quickly after loss
- Autotuning buffer space usage

Dijkstra, SC2002

148

## The Future of TCP/IP cont.

- Limited Transmit, RFC 3042 – open window on duplicate ACKs (Proposed Standard Jan 2001)
- Explicit Congestion Notification (ECN)
  - RFC 3168, Sep 2001
- Modifications to prevent “cheating”
- Kick-starting TCP after timeouts

Dijkstra, SC2002

149

## Increased Initial Windows

draft-ietf-tsvwg-initwin-04.txt

- Allows ~4KB initial window rather than one or two segments
  - $\min(4 \cdot \text{MSS}, \max(2 \cdot \text{MSS}, 4380 \text{ bytes}))$
- Proposed Standard Aug 28, 2002.

Dijkstra, SC2002

150

## Appropriate Byte Counting

draft-allman-tcp-abc-03.txt

- When an ACK is received, increase cwin based on the *number of new bytes ACK'd*
- Prevents receiver from “cheating” and making the sender open cwin too quickly
  - e.g. receiver ACKs every byte
- Increases by at most  $2 \times \text{MSS}$  bytes per ACK
  - To avoid bursts when one ACK covers a huge number of bytes

Dijkstra, SC2002

151

## Quick-Start

draft-amit-quick-start-01.txt

- IP option in the TCP SYN specified desired initial sending rate
  - Routers on the path decrement a TTL counter and decrease initial sending rate if necessary
- If all routers participated, receiver tells the sender the initial rate in the SYN+ACK pkt
- The sender can set cwin based on the rtt of the SYN and SYN+ACK packets

Dijkstra, SC2002

152

# Limited Slow-Start

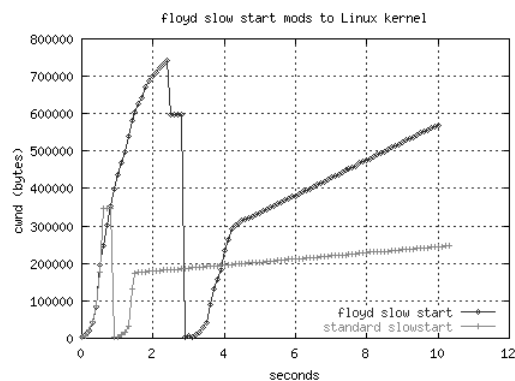
[www.icir.org/floyd/hstcp.html](http://www.icir.org/floyd/hstcp.html)

- In slow-start, the congestions window (cwin) doubles each round trip time
- For large cwin, this doubling can cause massive packet loss (and network load)
- Limited slow-start adds *max\_ssthresh* (proposed value of 100 MSS)
- Above *max\_ssthresh* cwin opens slower, never bursts more than 100 MSS  
$$cwin += (0.5 * \text{max\_ssthresh} / cwin) * \text{MSS}$$

Dijkstra, SC2002

153

## Limit Slow-Start Example



Tom Dunigan, ORNL

Dijkstra, SC2002

154

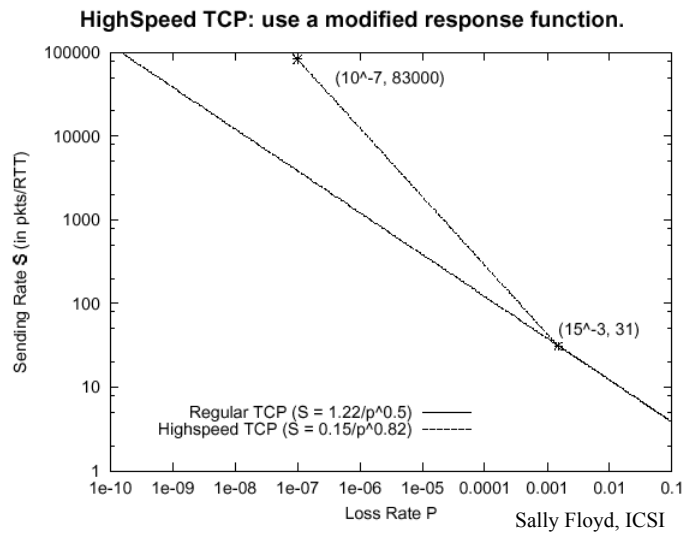
# HighSpeed TCP

[www.icir.org/floyd/hstep.html](http://www.icir.org/floyd/hstep.html)

- Changes the AIMD parameters
- Identical to standard TCP for loss rates below  $10^{-3}$  for fairness ( $cwin \leq 38$ )
- Allows  $cwin$  to reach 83000 segments for  $10^{-7}$  loss rates
  - Good for 10 Gbps over 100 msec rtt
  - Std TCP would be limited to  $\sim 440$  Mbps

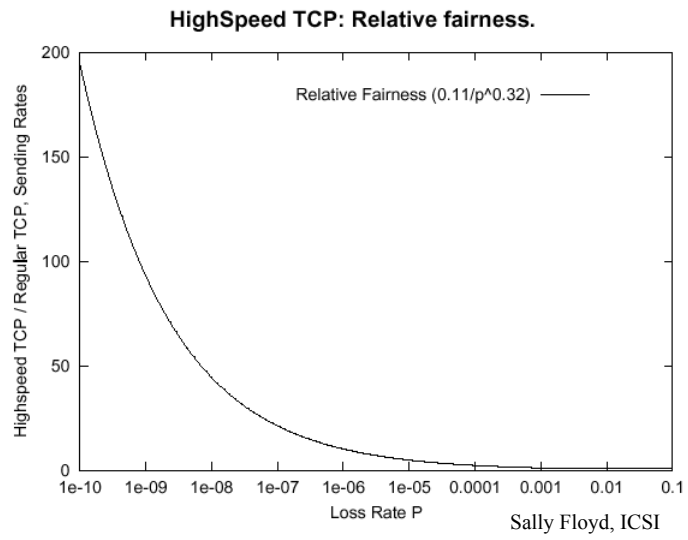
Dijkstra, SC2002

155



Dijkstra, SC2002

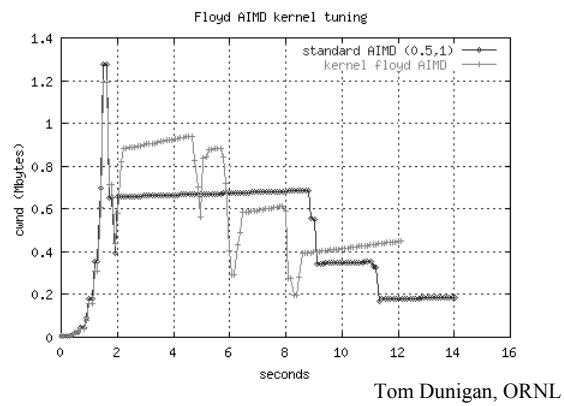
156



Dijkstra, SC2002

157

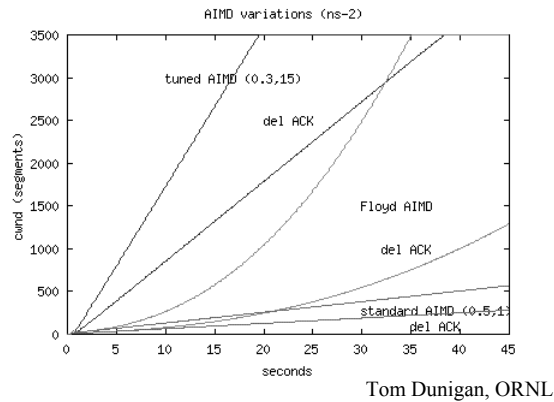
## Standard vs. HighSpeed TCP



Dijkstra, SC2002

158

# Cwin Opening Comparison



Dijkstra, SC2002

159

# IPv4 to IPv6 Changes

Vers 4	IHL	Type of Service	Total Length	
Identification			Flags	Frag Offset
Time to Live	Protocol		Header Checksum	
Source Address				
Destination Address				
IP Options				

v4 Header = 20 Bytes + Options  
v6 Header = 40 Bytes

Vers 6	Traffic Class	Flow Label		
Payload Length		Next Hdr	Hop Limit	
Source Address				
Destination Address				

Dijkstra, SC2002

160



# Multi-Protocol Label Switching (MPLS)

- Adds switched (layer 2) paths to below IP
  - Useful for traffic engineering, VPN's, QoS control, high speed switching
- IP packets get wrapped in MPLS frames and “labeled”
- MPLS routers switch the packets along Label Switched Paths (LSP's)
- Being generalized for optical switching

Dijkstra, SC2002

161

## Review

- Network capacity vs. speed
- Importance of window and buffer sizes
- How TCP throughput depends on delay, loss, packet size
- How to use ping, traceroute, treno, etc.
- Looking deeper for problems
- TCP/IP keeps evolving

Dijkstra, SC2002

162

## Recommended Resources

- Richard W. Stevens' books
  - TCP/IP Illustrated, ISBN 0-201-63346-9
  - <http://www.kohala.com/start/>
- Host performance tuning details
  - [http://www.psc.edu/networking/perf\\_tune.html](http://www.psc.edu/networking/perf_tune.html)
- CAIDA Internet Measurement Tool Taxonomy
  - <http://www.caida.org/tools/>

Dijkstra, SC2002

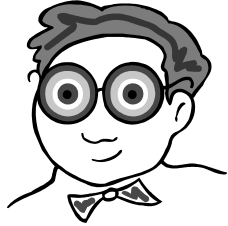
163

## Recommended Resources

- Iperf for TCP and UDP throughput testing
  - <http://dast.nlanr.net/Projects/Iperf/>
- Testrig for TCP traces
  - <http://ncne.nlanr.net/research/tcp/testrig/>
- Web 100
  - <http://www.web100.org/>

Dijkstra, SC2002

164



Thank You!



Phillip Dykstra  
WareOnEarth Communications Inc.  
2109 Mergho Impasse  
San Diego, CA 92110  
[phil@sd.wareonearth.com](mailto:phil@sd.wareonearth.com)  
619-574-7796